universität wien

DISSERTATION / DOCTORAL THESIS

Titel der Dissertation / Title of the Doctoral Thesis

"Mathematical Analysis of Deep Learning with Applications to Kolmogorov Equations"

verfasst von / submitted by

Julius Konstantin Berner, BSc MSc

angestrebter akademischer Grad / in partial fulfillment of the requirements for the degree of

Doktor der Naturwissenschaften (Dr. rer. nat.)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 796 605 405
Dissertationsgebiet lt. Studienblatt / field of study as it appears on the student record sheet:	Mathematik
Betreut von / Supervisor:	Univ.-Prof. Dr. Philipp Grohs

Mathematical Analysis of Deep Learning with Applications to Kolmogorov Equations

Julius Berner
University of Vienna

Abstract

This thesis comprises a series of publications that contribute to the emerging field of mathematical analysis of deep learning. The term deep learning refers to machine learning methods that use gradient-based optimization techniques to fit the parameters of deep neural networks to given data. Over the past decade, such approaches have catalyzed unprecedented advances across a wide range of applications. While a comprehensive mathematical explanation for their success remains elusive, this work provides fundamental insights that improve the theoretical understanding of deep learning. To facilitate a rigorous analysis, we focus on learning problems with known regularity properties, as frequently encountered in the context of differential equations. Specifically, we analyze deep learning algorithms for the numerical solution of a class of partial differential equations, known as Kolmogorov equations, employing representations based on stochastic differential equations. It is demonstrated that empirical risk minimization over deep neural networks efficiently approximates the solutions of families of Kolmogorov equations, with both the size of the neural networks and the number of samples scaling only polynomially in the underlying dimension. Furthermore, we introduce variance-reduced loss functions and identify settings in which local minima of corresponding optimization problems are nearly optimal. On the other hand, we also address the shortcomings of deep learning and establish fundamental constraints on learning neural networks from samples. Extensive numerical experiments corroborate the potential of deep learning to overcome the curse of dimensionality while revealing its inherent limitations. This comprehensive investigation contributes toward principled and reliable applications of deep learning in the natural sciences.

Zusammenfassung

Diese Arbeit umfasst eine Reihe an Publikationen, welche einen Beitrag zum aufkommenden Gebiet der mathematischen Analyse des tiefen Lernens leisten. Der Begriff „tiefes Lernen“ (engl. deep learning) bezeichnet Methoden des maschinellen Lernens, bei denen gradientenbasierte Optimierungsverfahren eingesetzt werden, um die Parameter von tiefen neuronalen Netzen an gegebene Daten anzupassen. In den letzten zehn Jahren haben solche Ansätze in einer Vielzahl von Anwendungen zu beispiellosen Fortschritten geführt. Während eine umfassende mathematische Erklärung für diesen Erfolg weiterhin aussteht, liefert die vorliegende Arbeit grundlegende Erkenntnisse, welche ein besseres theoretisches Verständnis des tiefen Lernens ermöglichen. Um eine rigorose Analyse zu ermöglichen, konzentrieren wir uns auf Lernprobleme mit bekannten Regularitätseigenschaften, wie sie oft im Kontext von Differentialgleichungen vorkommen. Insbesondere analysieren wir Algorithmen aus dem Bereich des tiefen Lernens für die numerische Lösung einer Klasse partieller Differentialgleichungen, bekannt als Kolmogorov-Gleichungen, unter Verwendung von Repräsentationen, welche auf stochastischen Differentialgleichungen basieren. Es wird gezeigt, dass empirische Risikominimierung über tiefe neuronale Netze die Lösungen von Familien von Kolmogorov-Gleichungen effizient approximiert, wobei sowohl die Größe der neuronalen Netze als auch die Anzahl an Datenpunkten nur polynomiell in der zugrunde liegenden Dimension skaliert. Darüber hinaus führen wir varianzreduzierte Verlustfunktionen ein und identifizieren Bedingungen, unter denen lokale Minima der entsprechenden Optimierungsprobleme nahezu optimal sind. Andererseits gehen wir auch auf die Unzulänglichkeiten des tiefen Lernens ein und stellen grundlegende Schranken für das Lernen neuronaler Netze aus Daten auf. Ausführliche numerische Experimente bestätigen das Potenzial des tiefen Lernens, den Fluch der Dimensionalität zu überwinden, wobei zugleich dessen inhärente Grenzen deutlich werden. Diese umfassende Untersuchung trägt zu fundierten und verlässlichen Anwendungen von tiefem Lernen in den Naturwissenschaften bei.

Acknowledgment

I would like to express my heartfelt appreciation to everyone who has accompanied me on my journey toward obtaining a PhD. They have made this time an unforgettable and deeply enriching experience, and without their support and encouragement, this work would not have come to fruition.

I am particularly grateful to Philipp Grohs, who has been a constant pillar of support since my master's studies. He introduced me to the captivating world of scientific research, sparked my passion for delving into the mathematical aspects of deep learning, and steered me toward intriguing and promising research avenues. His guidance and the excellent research environment in his group have been instrumental in helping me reach this important milestone.

Additionally, I would like to express my gratitude to Gitta Kutyniok, Felix Voigtlaender, Philipp Petersen, Arnulf Jentzen, Dennis Elbrächter, Lorenz Richter, and Markus Dablander for our fruitful collaborations that have culminated in the papers featured in this work. I cherished our engaging scientific discussions, through which I acquired a wealth of knowledge. Such experiences have reinforced my conviction that collaborations are indispensable for both scientific progress and personal growth.

I wish to extend my thanks to my remarkable colleagues at the office, who have significantly contributed to a nurturing and enjoyable work atmosphere. Our "uncountable" lunch and coffee breaks, along with various leisure activities, have greatly bolstered my motivation. Special recognition goes to my exceptional colleagues Pavol Harar and Dennis Elbrächter, with whom I had the privilege of sharing the "probably approximately coolest" (PAC) office in the known universe. I am also humbled to have had the opportunity to work with outstanding supervisors and colleagues during my internships at Meta and NVIDIA. I eagerly anticipate our paths crossing again in the future.

Moreover, I am truly fortunate to have the most incredible friends by my side. Although many of them did not engage in extensive mathematical discussions with me, they greatly enriched my life in other aspects, providing the necessary balance for pursuing a PhD.

Last but certainly not least, my utmost gratitude is undoubtedly reserved for my family, whose unwavering support and encouragement have been invaluable throughout my endeavors. Words cannot adequately convey the extent of my appreciation toward them.

Contents

Preamble

Publications

Appendix

Preamble

1 Introduction

> *"In short, success in creating AI, could be the biggest event in the history of our civilization. But it could also be the last, unless we learn how to avoid the risks."*
>
> *Stephen Hawking, 2016*

In recent years, the field of *artificial intelligence* (AI) has experienced unprecedented advances in areas such as computer vision, speech, natural language processing, and the natural sciences. Machine learning algorithms have been able to solve complex tasks that were previously unattainable with conventional, explicitly constructed algorithms. This progress is largely driven by the emergence of *deep learning*, a machine learning technique based on *deep neural networks*.

Loosely inspired by the functioning of the human brain, neural networks are models composed of interconnected *artificial neurons*, as shown in Figure 1.1. The first computationally applicable artificial neurons were already developed decades ago (McCulloch and Pitts, 1943; Rosenblatt, 1958). However, deep learning only gained significant traction following a breakthrough performance in an annual image classification challenge in 2012 (Krizhevsky, Sutskever, and Hinton, 2012).

The late rise of neural networks can be attributed to the development of specialized *architectures*, i.e., arrangements of artificial neurons. A significant change among architectures has been the transition from parallel to layered arrangements of neurons, resulting in so-called *deep* architectures. The breakthrough was also facilitated by the rapidly growing amount of available computational resources and digitally stored data. Substantial engineering efforts coupled with advances in high-performance computing on *graphics processing units* (GPUs) made it possible to train deep neural networks on massive amounts of data using gradient-based optimization techniques.

Despite their remarkable performance, deep learning algorithms, which are predominantly data-driven, remain largely opaque and lack theoretical guarantees. While a considerable amount of mathematical research on neural networks had been conducted before their empirical success, the paradigm shift and the astonishing performance of deep learning have raised a number of new questions and

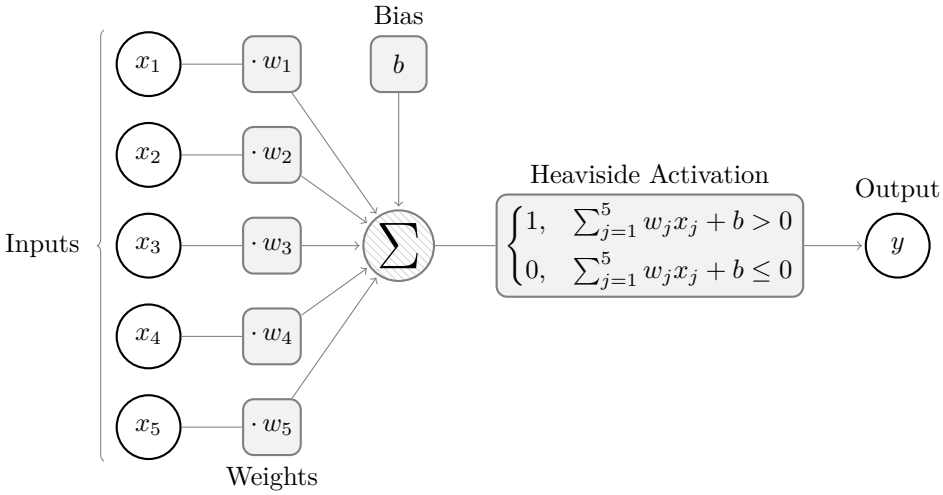


Figure 1.1: Computational graph of an artificial neuron with Heaviside activation function and parameters $w_1, \dots, w_5, b \in \mathbb{R}$. This represents a model of a biological neuron, which outputs an action potential, i.e., $y = 1$, at its *axon* if the weighted sum $\sum_{j=1}^d w_j x_j$ of the input signals $(x_j)_{j=1}^d$ from *dendritic spines* surpasses a certain threshold $-b$, see also Géron (2017).

challenges. This thesis aims to formulate these questions and provides answers and insights for selected questions and applications.

Particular emphasis is placed on the application of deep learning to scientific tasks with underlying physical models that can be mathematically formalized. Such models are often governed by *partial differential equations* (PDEs), arguably the most important modeling tool in the natural sciences. Despite a long history of research in the area of PDEs, the numerical solution of many highly relevant problems remains challenging or even completely out of reach. Recently, however, deep learning has also led to major breakthroughs in this area, opening up unimagined possibilities for practitioners. The present work shows how neural networks can be used to numerically solve certain classes of PDEs and demonstrates that the resulting algorithms can overcome complexity barriers of classical methods.

In summary, this work aims to define and summarize the emerging field of *mathematical analysis of deep learning*. It presents numerical algorithms based on deep learning and contributes toward their theoretical understanding. The primary focus is on applications in the context of PDEs, in particular so-called *Kolmogorov equations*, highlighting the potential of deep learning in solving complex scientific problems and ultimately advancing our understanding of the natural world.

2 Preliminaries

We start by giving a slightly abstract definition of deep learning and refer to Chapter I for further details. In a *learning problem*, one seeks a model which best explains observations from an underlying, typically unknown, data distribution. We assume that the data is represented by a $\mathcal{Z}$-valued random variable Z and that the set of *admissible models* is given by $\mathcal{M}$. We measure the performance of a model $f \in \mathcal{M}$ on a realization $z \in \mathcal{Z}$ using a *loss function* $\mathcal{L}\colon \mathcal{M} \times \mathcal{Z} \to [0, \infty)$. The corresponding learning problem[1]

$$\min_{f \in \mathcal{M}} \mathbb{E}[\mathcal{L}(f, Z)] \tag{2.1}$$

asks for a minimizer $f^* \in \mathcal{M}$ of the expected loss, the so-called *risk*,

$$\mathcal{E}(f) \coloneqq \mathbb{E}[\mathcal{L}(f, Z)], \quad f \in \mathcal{M}.$$

For instance, in a *regression task with quadratic loss function* and data $Z \coloneqq (X, Y)$ taking values in $\mathcal{Z} \coloneqq \mathcal{X} \times \mathbb{R}$, we have the loss function

$$\mathcal{L}_{\text{quadr}}(f, Z) \coloneqq (Y - f(X))^2, \quad f \in \mathcal{M}, \tag{2.2}$$

where we can choose $\mathcal{M}$ to be the space of measurable functions $f\colon \mathcal{X} \to \mathbb{R}$. The goal is to find a function $f \in \mathcal{M}$ that predicts the *labels* Y of given *features* X. In the case where Y has finite variance, the *tower property* of the conditional expectation ensures that

$$\mathbb{E}[\mathcal{L}_{\text{quadr}}(f, Z)] = \mathbb{E}[(\mathbb{E}[Y|X] - f(X))^2] + \mathbb{E}[(Y - \mathbb{E}[Y|X])^2]. \tag{2.3}$$

This decomposition shows that the optimal solution $f^* \in \mathcal{M}$ to the regression task satisfies almost surely that

$$f^*(X) = \mathbb{E}[Y|X]. \tag{2.4}$$

However, most learning problems as defined in (2.1) turn out to be infeasible. In particular, the data Z is generally unknown and one is only given realizations of samples $(Z^{(i)})_{i=1}^m$, typically assumed to be mutually independent and drawn from the distribution of Z. Moreover, the set $\mathcal{M}$ might also be too large to be optimized

[1] We work on a filtered probability space satisfying the usual conditions and denote by $\mathbb{E}[X]$ and $\mathbb{V}[X]$ the expectation and variance of a random variable X. For details on measurability and regularity assumptions that guarantee well-defined learning problems, we refer to Chapter III.

over. A common approach to finding an approximate solution to the learning problem in (2.1) is to consider the *empirical risk minimization* problem

$$\min_{f\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(f, Z^{(i)}) \tag{2.5}$$

over a suitable *hypothesis set* $\mathcal{H} \subset \mathcal{M}$. In this approach, one picks a model $\hat{f} \in \mathcal{H}$ that minimizes the average loss on the given samples $(Z^{(i)})_{i=1}^m$, the so-called *empirical risk*

$$\hat{\mathcal{E}}(f) \coloneqq \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(f, Z^{(i)}), \quad f \in \mathcal{M}.$$

Then, we can also define the *best approximation* $f_{\mathcal{H}}^* \in \mathcal{H}$ in the hypothesis set $\mathcal{H}$ as a minimizer of

$$\min_{f\in\mathcal{H}} \mathbb{E}[\mathcal{L}(f, Z)].$$

In the context of deep learning, the hypothesis set is given by a parametric family

$$\mathcal{H} \coloneqq \mathcal{R}(\Theta) = \{\mathcal{R}(\theta)\colon \theta \in \Theta\} \tag{2.6}$$

with a suitable *parameter space* Θ and *realization map* $\mathcal{R}\colon \Theta \to \mathcal{M}$. Assuming that the mapping $\theta \mapsto (\hat{\mathcal{E}} \circ \mathcal{R})(\theta)$ is differentiable, the empirical risk minimization problem in (2.5) is typically tackled using variants of *gradient descent*. In their most basic form, gradient descent schemes are given by the parameter update

$$\theta^{(n+1)} \coloneqq \theta^{(n)} - \lambda \nabla(\hat{\mathcal{E}} \circ \mathcal{R})(\theta^{(n)}), \quad \theta^{(0)} \in \Theta, \tag{2.7}$$

where $\lambda \in (0, \infty)$ is some step size, also known as *learning rate*. In practice, one usually considers *stochastic gradient descent*, where only a random subset of the samples $(Z^{(i)})_{i=1}^m$, a so-called *mini-batch*, is used in the empirical risk $\hat{\mathcal{E}}$ for each update in (2.7). For a sufficiently large number of steps $N \in \mathbb{N}$, the model $\mathcal{R}(\theta^{(N)})$ serves as a candidate for an approximate solution to the learning problem, as illustrated in Figure 2.1.

In deep learning, the realization map $\mathcal{R}$ in (2.6) is generally defined as a composition of simple functions $\mathcal{R}^{(\ell)}\colon \Theta \to C(\mathbb{R}^{a_{\ell-1}}, \mathbb{R}^{a_\ell})$, so-called *layers*, such that[2]

$$\mathcal{R}(\theta) = \mathcal{R}^{(L)}(\theta) \circ \cdots \circ \mathcal{R}^{(1)}(\theta), \quad \theta \in \Theta, \tag{2.8}$$

[2]In view of (2.6), this definition implies that $\mathcal{H} \subset C(\mathbb{R}^{a_0}, \mathbb{R}^{a_L})$, implicitly assuming a suitable set of admissible models $\mathcal{M} \supset \mathcal{H}$.

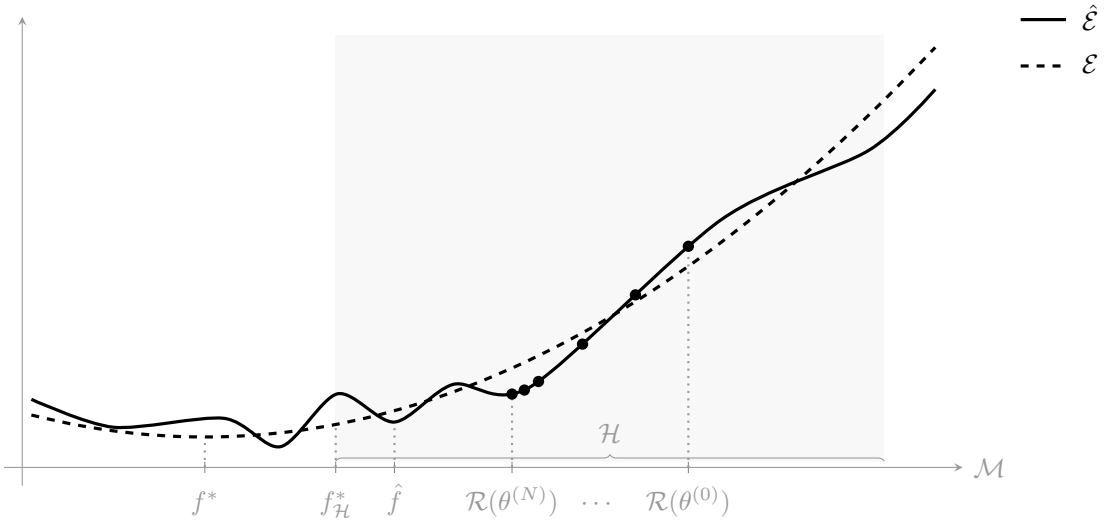


Figure 2.1: Illustration of the risk $\mathcal{E}$, the solution to the learning problem $f^* \in \operatorname{argmin}_{f\in\mathcal{M}} \mathcal{E}(f)$, the hypothesis set $\mathcal{H} \subset \mathcal{M}$, the best approximation $f^*_{\mathcal{H}} \in \operatorname{argmin}_{f\in\mathcal{H}} \mathcal{E}(f)$, the empirical risk $\hat{\mathcal{E}}$ (for a given realization of samples), the empirical risk minimizer $\hat{f} \in \operatorname{argmin}_{f\in\mathcal{H}} \hat{\mathcal{E}}(f)$, and a sequence of models $(\mathcal{R}(\theta^{(n)}))_{n=0}^N$ obtained by applying gradient descent to the mapping $\theta \mapsto (\hat{\mathcal{E}} \circ \mathcal{R})(\theta)$.

where $a = (a_\ell)_{\ell=0}^L \in \mathbb{N}^{L+1}$ is a given *architecture* with *depth* $L \in \mathbb{N}$. Models of the form $\mathcal{R}(\theta)$, $\theta \in \Theta$, are then commonly referred to as *neural networks*[3]. Based on analytic expressions of the derivatives of the simple functions in (2.8), the *chain rule* allows for an efficient and accurate evaluation of the gradients $\nabla(\hat{\mathcal{E}} \circ \mathcal{R})(\theta^{(n)})$ needed in (2.7) on a computer by means of *automatic differentiation*, a specific form of which is often called the *backpropagation algorithm* (Rumelhart, Hinton, and Williams, 1986).

We mostly focus on classical, *fully-connected feed-forward neural networks*, where the outputs of the simple functions $\mathcal{R}^{(\ell)}$ are affine-linear mappings composed with univariate functions which are applied componentwise. Specifically, we choose

$$\Theta \subset \bigtimes_{\ell=1}^{L} \left(\mathbb{R}^{a_\ell \times a_{\ell-1}} \times \mathbb{R}^{a_\ell}\right),$$

[3]By common abuse of notation, we use the term neural network to denote functions of the form $\mathcal{R}(\theta)$, as defined in (2.8). However, it is important to note that typical realization maps $\mathcal{R}$ are not injective. In order to talk about the *parametrization* $\theta \in \Theta$ of a neural network $\mathcal{R}(\theta)$, one would need to define the neural network as a tuple $(\mathcal{R}, \theta)$.

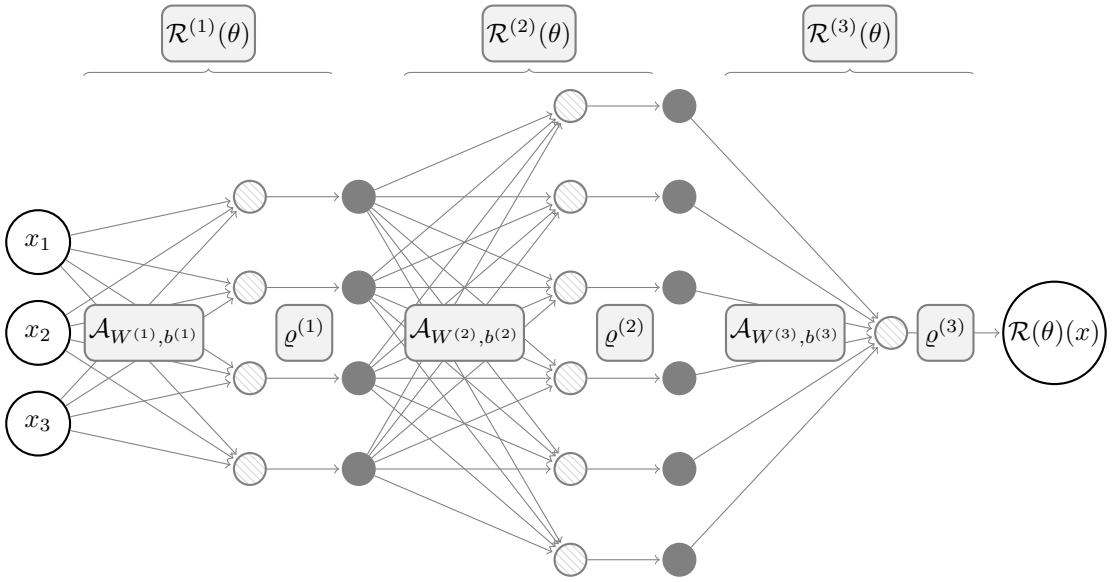


Figure 2.2: Computational graph of a fully connected feedforward neural network $\mathcal{R}(\theta)\colon \mathbb{R}^3 \to \mathbb{R}$ with architecture $a = (3, 4, 6, 1)$, depth 3, and parameters $\theta = ((W^{(\ell)}, b^{(\ell)}))_{\ell=1}^{3}$. In accordance with Figure 1.1, we refer to $W^{(\ell)}$ and $b^{(\ell)}$ as the *weights* and *biases* in the ℓ-th layer. We call $\mathcal{R}(\theta)$ a *ReLU network* if $\varrho^{(1)}(x) = \varrho^{(2)}(x) = \max\{x, 0\}$ and $\varrho^{(3)}(x) = x$ for $x \in \mathbb{R}$.

and for every $\ell \in \{1, \dots, L\}$ we define

$$\mathcal{R}^{(\ell)}\colon \Theta \to C(\mathbb{R}^{a_{\ell-1}}, \mathbb{R}^{a_\ell}), \quad \theta = ((W^{(\ell)}, b^{(\ell)}))_{\ell=1}^{L} \mapsto \varrho^{(\ell)} \circ \mathcal{A}_{W^{(\ell)},b^{(\ell)}}.$$

In the above, the functions

$$\mathcal{A}_{W^{(\ell)},b^{(\ell)}}\colon \mathbb{R}^{a_{\ell-1}} \to \mathbb{R}^{a_\ell}, \quad x \mapsto W^{(\ell)}x + b^{(\ell)},$$

denote affine-linear mappings and, by common abuse of notation, the *activation functions* $\varrho^{(\ell)} \in C(\mathbb{R})$ are applied componentwise, see Figure 2.2. In this context, the number of parameters of the architecture $a = (a_\ell)_{\ell=0}^{L}$ is given by

$$\dim(\Theta) \coloneqq \sum_{\ell=1}^{L} a_\ell a_{\ell-1} + a_\ell$$

and a_ℓ denotes the *width*, i.e., the number of artificial neurons, in the ℓ-th layer. A frequently utilized activation function, which is also used for the majority of results in this thesis, is the so-called *Rectified Linear Unit* (ReLU) given by

$$\varrho^{(\ell)}(x) = \varrho_R(x) \coloneqq \max\{x, 0\}, \quad x \in \mathbb{R},$$

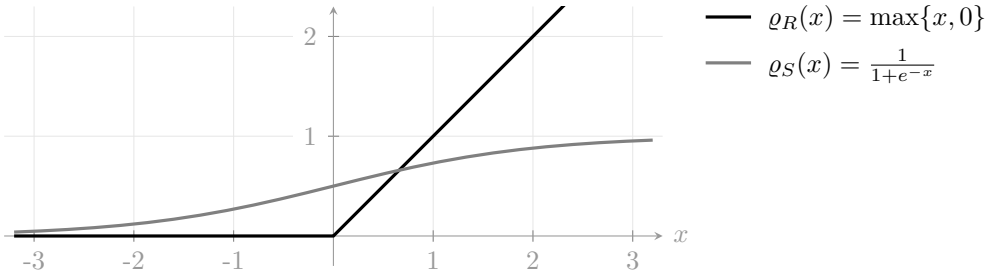


Figure 2.3: Two commonly used activation functions. In this thesis, we focus on the *ReLU activation function* ϱ_R. For binary *classification tasks*, where the labels Y take only two values, the *sigmoid activation function* ϱ_S can be used as activation function $\varrho^{(L)}$ in the last layer to interpret the output as the probability of a given class in the range $[0, 1]$.

as depicted in Figure 2.3. Depending on the desired output range, the last activation function $\varrho^{(L)}$ is often chosen differently. For instance, we will mostly use the identity, i.e., $\varrho^{(L)}(x) = x$. We further use the abbreviation *ReLU networks* to denote classical neural networks with ReLU activation functions in all layers except the last one.

Roughly speaking, the overarching question in the mathematical analysis of deep learning can be formulated as follows:

How well does a neural network $\mathcal{R}(\theta^{(N)})$ as defined in (2.8),
trained for N steps using a variant of gradient descent as in (2.7),
approximate the solution f^* of the learning problem in (2.1)?

Since this question cannot be answered in such generality, one often analyzes learning problems with additional prior knowledge, such as those arising in the context of PDEs. In this thesis, we repeatedly consider PDEs that admit stochastic representations based on *stochastic differential equations* (SDEs), often referred to as Kolmogorov equations. For simplicity, let us focus on linear, homogeneous Kolmogorov (backward) equations defined by

$$(\partial_t u + Fu)(x,t) = 0, \quad u(x,T) = \varphi(x), \quad (x,t) \in \mathbb{R}^d \times [0,T], \tag{2.9}$$

with spatial dimension $d \in \mathbb{N}$, terminal time $T \in (0, \infty)$, and differential operator F given by

$$Fu \coloneqq \frac{1}{2} \operatorname{Trace}\left(\sigma\sigma^\top \operatorname{Hess}_x(u)\right) + \mu \cdot \nabla_x u.$$

In the above,

$$\mu \in C(\mathbb{R}^d \times [0,T], \mathbb{R}^d) \quad \text{and} \quad \sigma \in C(\mathbb{R}^d \times [0,T], \mathbb{R}^{d\times d}) \tag{2.10}$$

are suitable *drift* and *diffusion coefficient functions*, $\varphi \in C(\mathbb{R}^d)$ is a given *terminal condition*, and we are interested in finding the corresponding solution[4] $u \in C^{2,1}(\mathbb{R}^d \times [0,T])$ of the PDE in (2.9).

The class of Kolmogorov equations covers a broad spectrum of PDEs, for which accurate and reliable solvers are of great importance to practitioners, see also Chapter V for an overview. For instance, such PDEs frequently appear in physics for the modeling of diffusion processes. Moreover, the class includes the *Black-Scholes equation* and extensions thereof, used for pricing financial derivative instruments. Via the *Hopf-Cole transform*, $u \mapsto -\log(u)$, Kolmogorov equations can also be connected to *Hamilton-Jacobi-Bellman equations*, prominent in the fields of *reinforcement learning* and *optimal control*.

Kolmogorov equations naturally lead to learning problems, as defined in (2.1). To see this, let ξ be a random variable with values on a compact subset $D \subset \mathbb{R}^d$ and define $S^\xi = (S_t^\xi)_{t\in[0,T]}$ to be a solution to the SDE

$$\mathrm{d}S_t^\xi = \mu(S_t^\xi, t)\,\mathrm{d}t + \sigma(S_t^\xi, t)\,\mathrm{d}B_t, \quad S_0^\xi = \xi, \tag{2.11}$$

with B denoting a standard d-dimensional Brownian motion. Under suitable conditions, see Chapter III, *Itô's formula* shows that almost surely it holds that

$$\begin{aligned}\varphi(S_T^\xi) &= u(S_T^\xi, T)\\ &= u(S_0^\xi, 0) + \int_0^T (\partial_t u + Fu)(S_t^\xi, t)\,\mathrm{d}t + \int_0^T \big(\sigma^\top \nabla_x u\big)(S_t^\xi, t)\cdot \mathrm{d}B_t\\ &= u(\xi, 0) + \int_0^T \big(\sigma^\top \nabla_x u\big)(S_t^\xi, t)\cdot \mathrm{d}B_t,\end{aligned} \tag{2.12}$$

where we used the fact that u solves the Kolmogorov equation in (2.9). Noting that, under mild regularity conditions, the stochastic integral in (2.12) has vanishing expectation conditioned on ξ, we obtain a version of the *Feynman-Kac* formula, i.e., it almost surely holds that

$$u(\xi, 0) = \mathbb{E}\big[\varphi(S_T^\xi)|\xi\big]. \tag{2.13}$$

In view of (2.4), this shows that the solution to the Kolmogorov equation in (2.9) evaluated at time $t = 0$, i.e.,

$$D \ni x \mapsto u(x, 0), \tag{2.14}$$

[4]For ease of presentation, we assume that there exists a unique strong solution $u \in C^{2,1}(\mathbb{R}^d \times [0,T])$ to the Kolmogorov equation in (2.9), which is twice continuously differentiable in the spatial coordinate $x \in \mathbb{R}^d$ and once continuously differentiable in the time coordinate $t \in [0,T]$. For conditions guaranteeing the existence and uniqueness of (viscosity) solutions u, we refer to the work by Hairer, Hutzenthaler, and Jentzen (2015).

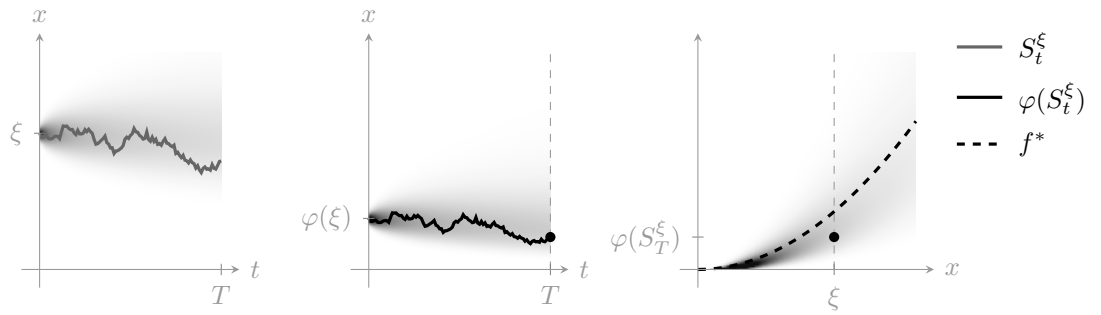


Figure 2.4: Realization of the solution S^ξ to the SDE in (2.11) and the associated data $(\xi, \varphi(S_T^\xi))$ of the learning problem for the Kolmogorov equation in (2.9), where $\sigma(x,t) \coloneqq \hat{\sigma}x$, $\mu(x,t) \coloneqq \frac{1}{2}\hat{\sigma}^2 x$, and $\varphi(x) \coloneqq \sum_{i=0}^n c_i x^i$ for $(x,t) \in \mathbb{R} \times [0,T]$ with $\hat{\sigma} = \frac{1}{4}$, $n = 2$, $c_0 = c_1 = 0$, $c_2 = \frac{1}{2}$, and $T = 1$. Using Itô's formula and the Laplace transform of a Gaussian random variable, one can show that $S_T^\xi = \xi e^{\hat{\sigma}B_T}$ and $f^*(x) = u(x,0) = \mathbb{E}\big[\varphi(S_T^\xi)|\xi = x\big] = \sum_{i=0}^n c_i e^{\frac{T}{2}(i\hat{\sigma})^2} x^i$, see Berner (2018).

is the unique[5] solution $f^* \in \mathcal{M} \coloneqq C(D)$ to the learning problem with data $Z = (\xi, \varphi(S_T^\xi))$ and quadratic loss function $\mathcal{L}_{\text{quadr}}$ as in (2.2), see also Figure 2.4. In Section 3.4, we will extend this framework to learn the whole time evolution of u, i.e., the mapping $D \times [0,T] \ni (x,t) \mapsto u(x,t)$.

For the above learning problem, we can easily generate (approximate) samples $(Z^{(i)})_{i=1}^m$ of the data $Z = (\xi, \varphi(S_T^\xi))$. Specifically, we can choose a suitable distribution for ξ, for instance, a uniform distribution on D, and simulate the SDE in (2.11) relying on standard numerical solvers. For instance, one can discretize the SDE solution $\hat{S}_{t_k}^\xi \approx S_{t_k}^\xi$ on a grid $0 = t_0 < \cdots < t_K = T$ with $K \in \mathbb{N}$ steps using the *Euler-Maruyama approximation* $(\hat{S}_{t_k}^\xi)_{k=0}^K$ given by

$$\hat{S}_{t_{k+1}}^\xi = \hat{S}_{t_k}^\xi + \mu(\hat{S}_{t_k}^\xi, t_k)(t_{k+1} - t_k) + \sigma(\hat{S}_{t_k}^\xi, t_k) \underbrace{(B_{t_{k+1}} - B_{t_k})}_{\sim \mathcal{N}(0, (t_{k+1}-t_k)\mathrm{I}_d)}, \quad \hat{S}_{t_0}^\xi = \xi, \tag{2.15}$$

see Chapter IV and Kloeden and Platen (1992) for convergence results and further numerical schemes. Given samples of the data Z, one can tackle the corresponding empirical risk minimization problem in (2.5) over a hypothesis set of neural networks $\mathcal{H} \coloneqq \mathcal{R}(\Theta)$ with input dimension $a_0 = d$ and output dimension $a_L = 1$ using variants of stochastic gradient descent. Given that we can control the errors

$$\mathcal{R}(\theta^{(N)}) \approx \hat{f} \approx f_{\mathcal{H}}^* \approx f^* = u(\cdot, 0), \tag{2.16}$$

[5] In general, the solution f^* is unique up to sets of measure zero w.r.t. the distribution of ξ. In the typical case where ξ follows a uniform distribution, the solution to the Kolmogorov equation, $D \ni x \mapsto u(x,0)$, is the unique *continuous* solution.

this results in an approximate solution $\mathcal{R}(\theta^{(N)})$ to the Kolmogorov equation, see also Figure 2.1.

We emphasize that such learning problems represent a rare scenario for real-world applications of deep learning. First, we have control over choosing the features ξ and can generate independent and identically distributed (i.i.d.) samples at will. Furthermore, we can leverage the underlying SDEs and PDEs to obtain a priori estimates on the regularity of the data $Z = (\xi, \varphi(S_T^\xi))$ and the solution to the learning problem $u(\cdot, 0)$. Such assumptions will be the basis for a rigorous mathematical analysis of the errors occurring in (2.16). In the following Chapter, we will summarize the publications included in this thesis and outline how they contribute to such an analysis.

3 Synopses of the Publications

This chapter presents the main concepts and findings of the publications that are contained in this thesis, see Table 3.1 for an overview.

	Title	**Reference**
Ch. I Sec. 3.1	*The Modern Mathematics of Deep Learning*	Berner, Grohs, Kutyniok, and Petersen (2022). *CUP*.
Ch. II Sec. 3.2	*Towards a regularity theory for ReLU networks–chain rule and global error estimates*	Berner, Elbrächter, Grohs, and Jentzen (2019). *SampTA*.
Ch. III Sec. 3.3	*Analysis of the Generalization Error: Empirical Risk Minimization over Deep Artificial Neural Networks Overcomes the Curse of Dimensionality in the Numerical Approximation of Black–Scholes Partial Differential Equations*	Berner, Grohs, and Jentzen (2020). *SIMODS*.
Ch. IV Sec. 3.4	*Numerically Solving Parametric Families of High-Dimensional Kolmogorov Partial Differential Equations via Deep Learning*	Berner, Dablander, and Grohs (2020). *NeurIPS*.
Ch. V Sec. 3.5	*Robust SDE-Based Variational Formulations for Solving Linear PDEs via Deep Learning*	Richter and Berner (2022). *ICML*.
Ch. VI Sec. 3.6	*How degenerate is the parametrization of neural networks with the ReLU activation function?*	Berner, Elbrächter, and Grohs (2019). *NeurIPS*.
Ch. VII Sec. 3.7	*Learning ReLU networks to high uniform accuracy is intractable*	Berner, Grohs, and Voigtlaender (2023). *ICLR*.

Table 3.1: Overview of the publications included in this thesis.

In Chapter 4, we will discuss the results and highlight potential extensions. To give a short outline, let us first state the research questions each of the publications attempts to answer:

I. The Modern Mathematics of Deep Learning

What are the open questions in the field of mathematical analysis of deep learning, and which results already exist?

II. Towards a Regularity Theory for ReLU Networks – Chain Rule and Global Error Estimates

What can be said about the approximation capabilities of deep neural networks, in particular with respect to Sobolev norms?

III. Analysis of the Generalization Error: Empirical Risk Minimization over Deep Artificial Neural Networks Overcomes the Curse of Dimensionality in the Numerical Approximation of Black–Scholes Partial Differential Equations

Given that the solution to the learning problem can be efficiently approximated by deep neural networks, can it also be learned by means of empirical risk minimization?

IV. Numerically Solving Parametric Families of High-Dimensional Kolmogorov Partial Differential Equations via Deep Learning

Can neural networks efficiently learn the solutions of parametric families of Kolmogorov equations?

V. Robust SDE-Based Variational Formulations for Solving Linear PDEs via Deep Learning

Can we reduce the variance of the gradients, which are used in gradient descent schemes for solving the corresponding empirical risk minimization problem?

VI. How Degenerate is the Parametrization of Neural Networks with the ReLU Activation Function?

How suited is the empirical risk minimization problem to the application of gradient descent based on prior knowledge about the regularity of the learning problem?

VII. Learning ReLU Networks to High Uniform Accuracy is Intractable

Do there exist fundamental lower bounds on the learnability of neural networks from samples?

3.1 The Modern Mathematics of Deep Learning

Over the past few decades, a substantial amount of research has been conducted to mathematically analyze machine learning methods and neural networks in particular. In this survey, we first summarize classical results in the field of learning theory. In particular, this includes the approximation and interpolation capabilities of so-called *shallow* neural networks with depth $L = 2$, the framework of *probably approximately correct* (PAC) learning based on *VC-dimensions* of hypothesis sets, and the convergence of variants of stochastic gradient descent for convex objectives. However, despite the plethora of results, recent applications of deep learning still pose a number of open questions. We further identify these newly emerging questions and define the field of mathematical analysis of deep learning. In addition, we give an overview of modern approaches, yielding partial answers to these questions. The questions concern the following topics:

1. **Generalization of overparametrized neural networks:** Successful applications of deep learning often use *overparametrized* neural networks $\mathcal{R}(\theta)$, $\theta \in \Theta$, where the number of parameters $\dim(\Theta)$ is significantly larger than the number m of given training samples $(Z^{(i)})_{i=1}^m$. Classical learning theory suggests that such expressive models $\mathcal{R}(\theta)$ would *overfit*, i.e., depict random fluctuation of the samples, and not generalize well, meaning that their risk $\mathcal{E}(\mathcal{R}(\theta))$ is significantly larger than their empirical risk $\hat{\mathcal{E}}(\mathcal{R}(\theta))$. Specifically, corresponding hypothesis sets $\mathcal{R}(\Theta)$ typically contain models that can interpolate the samples, and their VC-dimensions, which scale with the number of parameters, yield vacuous bounds on the generalization error. We summarize modern attempts to extend and reconcile classical learning theory for overparametrized models. For instance, we describe the *double descent* phenomenon suggesting that the risk of models can also decrease when their expressiveness surpasses the interpolation threshold. Moreover, we present complexity measures for neural network hypothesis sets based on norms of the parameters rather than their number. We also show that, under suitable assumptions, neural networks behave like linear models operating on nonlinear features (given by the so-called *neural tangent kernel*) in the infinite-width limit $a_1, \dots, a_{L-1} \to \infty$. Finally, we investigate the dependence of the generalization performance on the optimization scheme and, in particular, the *implicit bias* of gradient descent.

2. **Success of gradient descent:** In typical applications, the empirical risk minimization problem $\min_{\theta \in \Theta}(\hat{\mathcal{E}} \circ \mathcal{R})$ is non-convex (and non-smooth for activation functions such as the ReLU). Thus, one would expect that (stochastic) gradient descent gets stuck in local minima and does not converge to parameters $\theta^{(N)}$ which approximately minimize the empirical risk, i.e.,

$\hat{\mathcal{E}}(\mathcal{R}(\theta^{(N)})) \approx \hat{\mathcal{E}}(\hat{f})$. However, we present results showing that the *loss landscape*, i.e., the graph of $\theta \mapsto (\hat{\mathcal{E}} \circ \mathcal{R})(\theta)$, can be well-behaved for the use of gradient descent. For instance, we describe scenarios where local minima are almost optimal, see also Section 3.6, or where the empirical risk is only slightly increasing on a suitable path from a local to a global minimizer. We also show how the neural tangent kernel can be used to prove convergence results for overparametrized neural networks.

3. **Effects of different architectures:** Classical results primarily investigated the expressiveness of *shallow* feed-forward neural networks. Under mild assumptions on the activation function, the *universal approximation theorem* guarantees that depth $L = 2$ suffices to approximate any continuous function arbitrarily well on compact sets. However, commonly used architectures in practice are typically very deep. As a theoretical explanation, we present results showing that the *efficient* approximation of certain function classes, such as radial functions or smooth functions, indeed requires sufficiently deep architectures. Moreover, we list mathematical properties of commonly used types of neural networks, such as *convolutional*, *residual*, and *recurrent neural networks*, *U-Nets*, as well as *batch normalization* layers. The connections of neural networks to other iterative or hierarchical methods (e.g., the *scattering transform* or methods in *sparse coding*) further reveal potential invariances and sparsity in the (intermediate) outputs of neural networks.

4. **Applicability to high-dimensional problems:** The complexity of grid-based methods inevitably scales exponentially in the underlying dimension, a phenomenon often referred to as the *curse of dimensionality*. We present several settings where neural networks are able to overcome the curse of dimensionality, possibly explaining successful applications of deep learning to very high-dimensional problems in practice. These explanations build upon assumptions on the structure of the learning problem, such as underlying differential equations, stochastic representations, or the *manifold hypothesis*, which assumes that the features X of a regression task have a high likelihood of lying close to a low-dimensional manifold. The compositionality of neural networks then allows us to emulate and combine classical methods, such as *Monte-Carlo methods* and local *Taylor approximations* on manifolds, see also Sections 3.2 and 3.3. Finally, we mention deep learning methods for solving high-dimensional problems in the natural sciences, e.g., in the context of *inverse problems* and PDEs, see also Sections 3.4 and 3.5.

The paper presents selected approaches in more detail to provide the reader with fundamental ideas and intuition. As mentioned above, we will also shed light on these questions in the following sections.

3.2 Towards a Regularity Theory for ReLU Networks – Chain Rule and Global Error Estimates

This paper deals with the approximation capabilities of ReLU networks, i.e., how the architecture $a = (a_\ell)_{\ell=0}^L$ needs to be chosen to ensure that

$$\inf_{\theta\in\Theta} \|\mathcal{R}(\theta) - f^*\| \leq \varepsilon$$

for given $\varepsilon \in (0,\infty)$ and norm $\|\cdot\|$. While the solution to the learning problem f^* is typically unknown, we can utilize regularity properties of f^* and analyze the worst-case error over a corresponding function class. For instance, one might know a priori that f^* is contained in a ball in the *Sobolev space* $\mathcal{W}^{k,p}$, which consists of functions having weak derivatives up to order $k \in \mathbb{N}$ belonging to the Lebesgue space $\mathcal{L}^p$ with $p \in [1,\infty]$. Such estimates can, for instance, also be established for (weak) solutions of Kolmogorov equations (Evans, 2010, Section 7.1).

Generally speaking, neural networks exhibit optimal approximation capabilities for a variety of function classes (Elbrächter et al., 2021). The expressivity of neural networks is largely induced by the fact that compositions and linear combinations of neural networks can again be represented as a neural network. In contrast, classical, dictionary-based approximation methods only employ linear combinations of simple basis functions.

The compositionality of neural networks has, for instance, been used in a series of approximation results for ReLU networks originating from a construction of Yarotsky (2017). The main idea is to combine very simple functions to gradually build up approximations of complex functions. We start with the series representation

$$xy = \sum_{n=1}^{\infty} 4^{-n}\big(h_2^{\circ n}(x) + h_2^{\circ n}(y) - h_2^{\circ n}(x+y)\big), \quad x,y \in [0,1], \tag{3.1}$$

where $h_\delta^{\circ n}$ denotes the n-fold composition of the hat function

$$h_\delta \colon \mathbb{R} \to \mathbb{R}, \quad h_\delta(x) \coloneqq 2\varrho_R(x) - 4\varrho_R\left(x - \frac{\delta}{2}\right), \quad \delta \in (0,\infty), \tag{3.2}$$

depicted in Figure 3.1. From (3.2) it is obvious that h_δ and, by compositionality, $h_\delta^{\circ n}$ can be represented as ReLU networks. Together with the exponential decay of the series in (3.1), this implies that the multiplication operation, i.e., $(x,y) \mapsto xy$, can be approximated on bounded domains by ReLU networks with depth and number of parameters scaling only logarithmically in the reciprocal accuracy. Consequently, by using linear combinations and compositions, the same holds for polynomials, enabling efficient approximation of various types of regular functions.

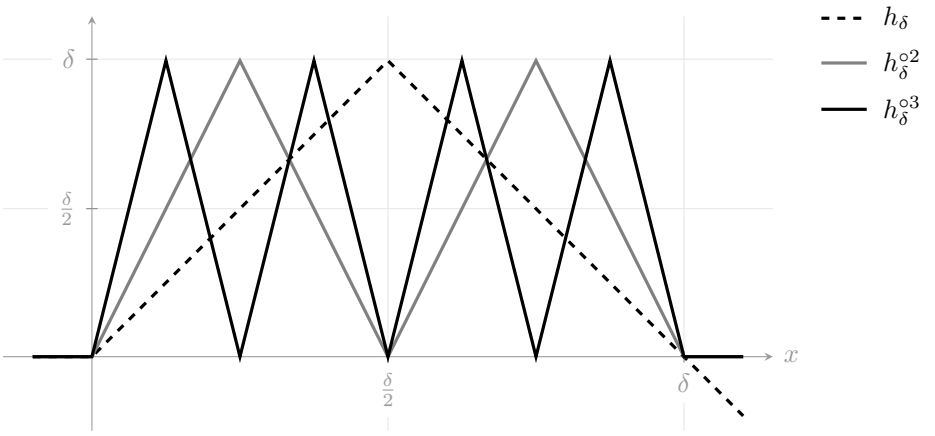


Figure 3.1: Composition of the hat function h_δ in (3.2) with itself.

To establish approximation results for the composition of functions f and g in such constructions, estimates on $\|\mathcal{R}(\theta_g) - g\|$ and $\|\mathcal{R}(\theta_f) - f\|$ are used to obtain a bound on

$$\|\mathcal{R}(\theta_g) \circ \mathcal{R}(\theta_f) - g \circ f\|. \tag{3.3}$$

While this has been successfully carried out for the $\mathcal{L}^\infty$-norm by Yarotsky (2017), we consider the stronger Sobolev norm $\|\cdot\| = \|\cdot\|_{\mathcal{W}^{1,\infty}}$, measuring the simultaneous uniform approximation of a function and its derivative. In particular, this requires estimating the derivative of $\mathcal{R}(\theta_g) \circ \mathcal{R}(\theta_f)$ in (3.3). When using classical neural networks with locally Lipschitz continuous activation functions $(\varrho^{(\ell)})_{\ell=1}^L$, the composition $\mathcal{R}(\theta_g) \circ \mathcal{R}(\theta_f)$ is almost everywhere differentiable. However, the chain rule can, in general, not be applied if $\varrho^{(\ell)}$ exhibits points of non-differentiability, such as the ReLU activation function. It might happen that the inner function maps sets of nonzero measure to points where the outer function is non-differentiable.

We circumvent this issue by defining a derivative of ReLU networks that coincides almost everywhere with the classical derivative and obeys a version of the chain rule. Moreover, we explain how this notion of derivative can lead to approximation results for smooth functions, e.g., with bounded $\mathcal{W}^{k,p}$-norm for $k \geq 2$ and $p \in [1, \infty]$, using a *partition of unity* and localized (averaged) *Taylor polynomials*, see also the concurrent work by Gühring, Kutyniok, and Petersen (2020).

Finally, we demonstrate how to construct neural networks $\mathcal{R}(\theta_f)$ *globally* approximating functions $f \in C^2(\mathbb{R}^d)$ with at most polynomially growing derivatives in the sense that

$$|\mathcal{R}(\theta_f)(x) - f(x)| \leq \varepsilon(1 + \|x\|_2^\nu), \quad x \in \mathbb{R}^d, \tag{3.4}$$

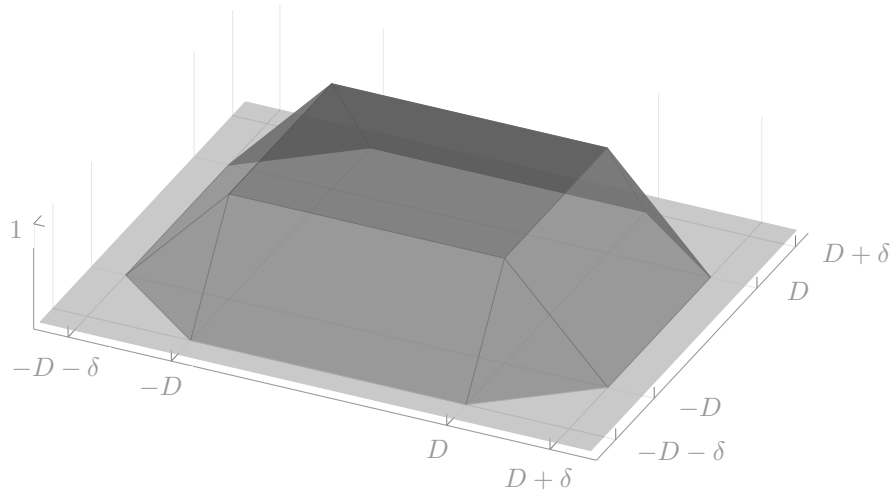


Figure 3.2: An approximation of the characteristic function $\mathbb{1}_{[-D,D]^2}$ by the ReLU network in (3.6) with $d = 2$.

and

$$\|\nabla\mathcal{R}(\theta_f)(x) - \nabla f(x)\|_2 \lesssim \varepsilon^\eta(1 + \|x\|_2^\nu), \quad x \in \mathbb{R}^d, \tag{3.5}$$

for some $\nu, \eta \in (0, \infty)$. The main idea is to adapt a neural network, which approximates f on a sufficiently large domain, to be zero outside that domain. This can be achieved using an approximative multiplication with the ReLU network

$$x \mapsto \varrho_R\left(\frac{1}{\delta}\sum_{i=1}^{d}\big(\varrho_R(x_i + D + \delta) - \varrho_R(x_i + D) - \varrho_R(x_i - D)\big) - (d-1)\right), \tag{3.6}$$

which approximates the characteristic function $\mathbb{1}_{[-D,D]^d}$ of the hypercube $[-D, D]^d$ with $D \in (0, \infty)$ for sufficiently small $\delta \in (0, \infty)$, as depicted in Figure 3.2. The next sections show that estimates of the form (3.4) and (3.5) are, for instance, demanded in the context of Kolmogorov equations.

3.3 Analysis of the Generalization Error: Empirical Risk Minimization over Deep Artificial Neural Networks Overcomes the Curse of Dimensionality in the Numerical Approximation of Black–Scholes Partial Differential Equations

The previous paper focused on estimating how well the solution to a learning problem f^* can be approximated by a hypothesis set of neural networks $\mathcal{H}$ based on regularity assumptions on f^*. This paper goes one step further and studies the

error between the empirical risk minimizer $\hat{f}$ over $\mathcal{H}$ and f^*. For a regression task as in (2.2), we can leverage the decomposition in (2.3) to show that[1]

$$\|\hat{f} - f^*\|_{\mathcal{L}^2(\mathbb{P}_X)} = \underbrace{\mathcal{E}(\hat{f}) - \mathcal{E}(f^*_{\mathcal{H}})}_{\textit{generalization error}} + \underbrace{\|f^*_{\mathcal{H}} - f^*\|_{\mathcal{L}^2(\mathbb{P}_X)}}_{\textit{approximation error}}, \tag{3.7}$$

see also Figure 2.1. For f^* that is sufficiently smooth, the approximation error could be tackled with tools outlined in the last section, e.g., using local Taylor approximation. However, note that such approximation results naturally underlie the curse of dimensionality. This paper focuses on the specific application of learning solutions $f^* = u(\cdot, 0)$ to Kolmogorov equations as defined in (2.14). In this case, the features $X = \xi$ are typically chosen to follow a uniform distribution such that the $\mathcal{L}^2(\mathbb{P}_X)$-norms in (3.7) correspond to standard (scaled) $\mathcal{L}^2(D)$-norms. Leveraging the structure of the underlying PDE and the Feynman-Kac formula in (2.13), we refine the estimates of Grohs, Hornung, et al. (2020) to establish approximation results that only scale polynomially in the spatial dimension d of the corresponding Kolmogorov equation.

Specifically, for time-homogeneous, affine-linear coefficient functions μ and σ in (2.10), one can prove the existence of random variables W and b, such that for every $x \in \mathbb{R}^d$ it holds almost surely that

$$S^x_T = \mathcal{A}_{W,b}(x) = Wx + b, \tag{3.8}$$

where $S^x = (S^x_t)_{t\in[0,T]}$ denotes the solution to the SDE in (2.11) with initial condition $x \in \mathbb{R}^d$. We further assume that the terminal condition φ is bounded and can be approximated by a neural network $\mathcal{R}(\theta_\varphi)$ in the sense of (3.4). Now, let $((W^{(i)}, b^{(i)}))_{i=1}^n$ be independent samples drawn from the distribution of (W, b) and define the (random) function

$$f \coloneqq \frac{1}{n}\sum_{i=1}^{n} \mathcal{R}(\theta_\varphi) \circ \mathcal{A}_{W^{(i)},b^{(i)}}. \tag{3.9}$$

Then, we can use the Feynman-Kac formula in (2.13), combined with the representation in (3.8) and uniform Monte Carlo estimates, to establish the bound

$$\mathbb{E}\big[\|f - u(\cdot,0)\|_{\mathcal{L}^2}\big] \le \varepsilon \big\|1 + \mathbb{E}\big[\|\mathcal{A}_{W,b}\|_2^\nu\big]\big\|_{\mathcal{L}^2} + \frac{1}{\sqrt{n}}\big\|\mathbb{V}\big[\mathcal{R}(\theta_\varphi)\circ\mathcal{A}_{W,b}\big]^{\frac{1}{2}}\big\|_{\mathcal{L}^2}. \tag{3.10}$$

The monotonicity of the expectation implies that there is a realization of f in (3.9) such that the bound in (3.10) still holds. As any realization of f can be represented

[1] For a measurable function $f\colon \mathcal{X} \to \mathbb{R}$, we define $\|f\|^p_{\mathcal{L}^p(\mathbb{P}_X)} \coloneqq \int |f|^p \,\mathrm{d}\mathbb{P}_X$, $p \in [1,\infty)$, and $\|f\|_{\mathcal{L}^\infty(\mathbb{P}_X)} \coloneqq \inf\{c \in [0,\infty)\colon |f(X)| \le c \text{ almost surely}\}$, where $\mathbb{P}_X$ denotes the distribution of the features X. In the following, we further abbreviate $\|\cdot\|_{\mathcal{L}^p} \coloneqq \|\cdot\|_{\mathcal{L}^p(\mathbb{P}_X)}$.

as a neural network, approximation of the terminal condition φ without the curse of dimensionality carries over to the solution $u(\cdot, 0)$ of the Kolmogorov equation.

Based on such bounds on the approximation error in (3.7), the paper develops a general framework to establish bounds on the generalization error, which are also not succumbed to the curse of dimensionality. Assuming that the functions in $\mathcal{H}$ are uniformly bounded, we can show that

$$\mathcal{E}(\hat{f}) - \mathcal{E}(f_{\mathcal{H}}^*) \leq 2r\big(\operatorname{Lip}(\mathcal{E}) + \operatorname{Lip}(\hat{\mathcal{E}}\,)\big) + 2\max_{k=1}^{K} |\mathcal{E}(f_k) - \hat{\mathcal{E}}(f_k)|, \tag{3.11}$$

where $(f_k)_{k=1}^K$ is a covering of $\mathcal{H}$ with radius $r \in (0, \infty)$ and Lip denotes the Lipschitz constant w.r.t. the $\|\cdot\|_{\mathcal{L}^\infty}$-norm. Note that, by definition, it holds that

$$\mathbb{E}\big[\hat{\mathcal{E}}(f_k)\big] = \mathcal{E}(f_k), \quad k = 1, \dots, K.$$

The last term in (3.11) can thus be bounded using a union bound and concentration inequalities. The size $K \in \mathbb{N}$ of a covering of neural network realizations $\mathcal{H} = \mathcal{R}(\Theta)$ with compact parameter space Θ is obtained from the size of a covering of Θ and estimates of the Lipschitz constant of the realization map $\mathcal{R}\colon \Theta \to \mathcal{H}$. The latter can be estimated by a layer-wise induction argument. In summary, we show that, with high probability over draws of data $(Z^{(i)})_{i=1}^m$, the number of samples m, needed to guarantee a given generalization error in (3.7), i.e.,

$$\mathcal{E}(\hat{f}) - \mathcal{E}(f_{\mathcal{H}}^*) \leq \varepsilon,$$

scales polynomially in the underlying dimension d and the reciprocal accuracy ε^{-1}, given that the size of the hypothesis set $\mathcal{H}$ required to achieve an approximation error of ε in (3.7) also scales polynomially in d and ε^{-1}.

Thus, we conclude that empirical risk minimization over deep neural networks can overcome the curse of dimensionality for the numerical solution of Kolmogorov equations with affine-linear coefficient functions whenever the terminal condition φ can be efficiently approximated. This assumption is, for instance, satisfied for Kolmogorov equations arising in option pricing problems from financial engineering, where typical terminal conditions φ can be exactly represented by ReLU networks.

For time-homogeneous, affine-linear coefficient functions μ and σ, every realization of S_T^x can be represented by an affine-linear function, see (3.8). However, we can also obtain approximations to realizations of S_T^x by emulating SDE discretization schemes, such as the Euler-Maruyama method in (2.15), with deep neural networks. Assuming sufficient regularity, this allows obtaining analogous approximation and generalization results for more general Kolmogorov equations if additionally μ and σ can be efficiently approximated in the sense of (3.4) and (3.5), see also Jentzen, Salimova, and Welti (2021) and Reisinger and Zhang (2020) for similar constructions.

3.4 Numerically Solving Parametric Families of High-Dimensional Kolmogorov Partial Differential Equations via Deep Learning

The results of the previous paper are supported by numerical experiments conducted by Beck, Becker, et al. (2021), which show that the solution to the Kolmogorov equation in (2.9) at time $t = 0$, i.e.,

$$D \ni x \mapsto u(x, 0),$$

can indeed be efficiently learned in high dimensions. We generalize this work by investigating the numerical solution of *parametric* families of Kolmogorov equations on the whole space-time domain $D \times [0, T]$. Specifically, we are given terminal conditions $(\varphi_\gamma)_{\gamma\in\Lambda}$ and coefficient functions $(\mu_\gamma)_{\gamma\in\Lambda}$ and $(\sigma_\gamma)_{\gamma\in\Lambda}$, parametrized by some parameter set $\Lambda \subset \mathbb{R}^p$ with $p \in \mathbb{N}$, and we want to numerically compute the solutions $(u_\gamma)_{\gamma\in\Lambda}$ of the corresponding parametrized family of Kolmogorov equations in (2.9). In other words, we want to learn the *parametric solution map*

$$\bar{u} \colon \mathbb{R}^d \times [0, T] \times \Lambda \to \mathbb{R}, \quad (x, t, \gamma) \mapsto u_\gamma(x, t), \tag{3.12}$$

which we assume to be continuous. To this end, let ξ, τ, and Γ be random variables distributed on compact space, time, and parameter domains D, $[0, T]$, and Λ. Similar to (2.12), Itô's formula shows that, under suitable conditions, it almost surely holds that

$$\varphi_\Gamma(S_T^X) = u_\Gamma(\xi, \tau) + \int_\tau^T \big(\sigma_\Gamma^\top \nabla_x u_\Gamma\big)(S_t^X, t) \cdot \mathrm{d}B_t, \tag{3.13}$$

where $X \coloneqq (\xi, \tau, \Gamma)$ and $S^X = (S_t^{\xi,\tau,\Gamma})_{t\in[0,T]}$ satisfies the SDE

$$\mathrm{d}S_t^X = \mu_\Gamma(S_t^X, t)\,\mathrm{d}t + \sigma_\Gamma(S_t^X, t)\,\mathrm{d}B_t, \quad S_\tau^X = \xi. \tag{3.14}$$

Using the vanishing expectation of the stochastic integral in (3.13) conditioned on X, the corresponding version of the Feynman-Kac formula in (2.13) states that

$$\mathbb{E}[\varphi_\Gamma(S_T^X)|X] = u_\Gamma(\xi, \tau)$$

almost surely. In other words, the optimal solution $f^* \in \mathcal{M} \coloneqq C(D \times [0, T] \times \Lambda)$ of the learning problem with data

$$Z = (X, Y) \coloneqq ((\xi, \tau, \Gamma), \varphi_\Gamma(S_T^{\xi,\tau,\Gamma})) \tag{3.15}$$

and quadratic loss function $\mathcal{L}_{\text{quadr}}$ is given by the parametric solution map

$$D \times [0, T] \times \Lambda \ni (x, t, \gamma) \mapsto \bar{u}(x, t, \gamma).$$

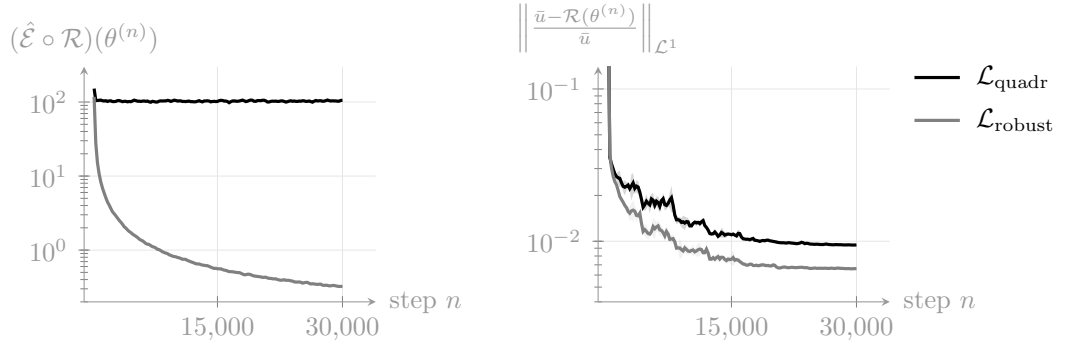


Figure 3.3: Average empirical risk and relative $\mathcal{L}^1$-error (and corresponding standard deviations over four random seeds in light gray) w.r.t. the number of gradient steps n when solving a parametric heat equation using the algorithm developed in Section 3.4 (in black). We follow the settings described in Chapter IV with dimension $d = 5$, batch-size 8192, and $N = 30000$ steps, resulting in $m = 8192N$ samples. In particular, we choose a parabolic terminal condition $\varphi_\gamma(x) = \|x\|_2^2$ and coefficient functions $\mu_\gamma(x,t) = 0$ and $\sigma_\gamma(x,t) = \gamma$, such that the reference solution is given by $f^*(x,t,\gamma) = u_\gamma(x,t) = \|x\|_2^2 + (T-t)\,\mathrm{Trace}(\gamma\gamma^\top)$ for $(x,t,\gamma) \in \mathbb{R}^d \times [0,T] \times \mathbb{R}^{d\times d}$. The loss function $\mathcal{L}_{\text{robust}}$ (in gray) will be presented in (3.22) in the next section.

One can now simulate independent samples $((X^{(i)}, Y^{(i)}))_{i=1}^m$ distributed according to (X, Y) in (3.15) and minimize the corresponding empirical risk

$$\hat{\mathcal{E}}(f) = \frac{1}{m}\sum_{i=1}^m \mathcal{L}_{\text{quadr}}(f, (X^{(i)}, Y^{(i)})) = \frac{1}{m}\sum_{i=1}^m (Y^{(i)} - f(X^{(i)}))^2, \quad f \in \mathcal{H}, \tag{3.16}$$

over a hypothesis set of neural networks $\mathcal{H} = \mathcal{R}(\Theta)$ with input dimension $a_0 = d+p+1$ and output dimension $a_L = 1$ using variants of stochastic gradient descent.

Specifically, we propose a neural network architecture inspired by *Multilevel Monte Carlo methods* (Berner, 2018; Giles, 2008), which significantly improves the numerical performance. In Figure 3.3, we present exemplary results for an implementation of this method. Our extensive experiments in Chapter IV indicate that such approaches are capable of numerically solving parametric families of Kolmogorov equations with only polynomial dependence on the input dimension. For selected examples, this is supported by theoretical bounds on the approximation and generalization errors based on the techniques obtained in the previous sections.

Using a trained neural network $\mathcal{R}(\theta^{(N)})$, satisfying that

$$\mathcal{R}(\theta^{(N)}) \approx \bar{u}, \tag{3.17}$$

one can easily compute partial derivatives of the form $\partial_\gamma \mathcal{R}(\theta^{(N)})$ by means of automatic differentiation. Assuming that the approximation in (3.17) also holds for the derivatives[2], this allows analyzing the sensitivity of the solution u_γ w.r.t. the parameter γ or fitting the parameter γ to a given data set using variants of gradient descent.

3.5 Robust SDE-Based Variational Formulations for Solving Linear PDEs via Deep Learning

This work aims to improve the method for learning solutions to (parametric families of) Kolmogorov equations presented in the previous section. More precisely, we propose variance-reduced versions of the loss function $\mathcal{L}_{\text{quadr}}$ used in the empirical risk $\hat{\mathcal{E}}$ in (3.16). To this end, we define the stochastic integral

$$\mathcal{I}_f \coloneqq \int_\tau^T \left(\sigma_\Gamma^\top(S_t^X, t)\nabla_x f(S_t^X, t, \Gamma)\right) \cdot \mathrm{d}B_t \tag{3.18}$$

for suitable $f \in \mathcal{M}$. For instance, we can choose $\mathcal{M}$ to be the space of continuous functions on $\mathbb{R}^d \times [0,T] \times \Lambda$ which are continuously differentiable in the spatial coordinate. Recalling Itô's formula in (3.13), it almost surely holds that

$$Y - \bar{u}(X) = \varphi_\Gamma(S_T^X) - u_\Gamma(\xi, \tau) = \mathcal{I}_{\bar{u}}. \tag{3.19}$$

Under suitable regularity assumptions, this establishes that for the optimal solution, i.e., $\mathcal{R}(\theta^*) = \bar{u}$, it holds that

$$\mathbb{V}\left[(\hat{\mathcal{E}} \circ \mathcal{R})(\theta^*)\right] = \frac{1}{m}\mathbb{V}\left[(Y - \mathcal{R}(\theta^*)(X))^2\right] = \frac{1}{m}\mathbb{V}\left[\mathcal{I}^2_{\mathcal{R}(\theta^*)}\right] \tag{3.20}$$

and

$$\mathbb{V}\left[\nabla(\hat{\mathcal{E}} \circ \mathcal{R})(\theta^*)\right] = \frac{4}{m}\mathbb{V}\left[\mathcal{I}_{\mathcal{R}(\theta^*)}\nabla_\theta \mathcal{R}(\theta^*)(X)\right]. \tag{3.21}$$

While the variances of the empirical risk and its gradient in (3.20) and (3.21) can be reduced by choosing a larger number of samples m, we observe that they are, in general, not vanishing at the optimum $\bar{u}$, see also Figure 3.4. Variants of gradient descent as in (2.7) rely on the gradient $\nabla(\hat{\mathcal{E}} \circ \mathcal{R})$ in (3.21) and might therefore not stick to the optimal parameters θ^*.

To counteract this issue, we propose to replace the quadratic loss function $\mathcal{L}_{\text{quadr}}$ in (3.16) with the loss function

$$\mathcal{L}_{\text{robust}}(f, Z) \coloneqq (Y - f(X) - \mathcal{I}_f)^2, \quad f \in \mathcal{M}, \tag{3.22}$$

[2]Such approximations can be proven to exist using the techniques described in Section 3.2. As an easy example, we note that ReLU networks with $\mathcal{O}(d^4 \log(d/\varepsilon)$ parameters can approximate the solution of the problem in Figure 3.3 up to precision ε in the $\mathcal{W}^{1,\infty}(D \times [0,T] \times \Lambda)$-norm.

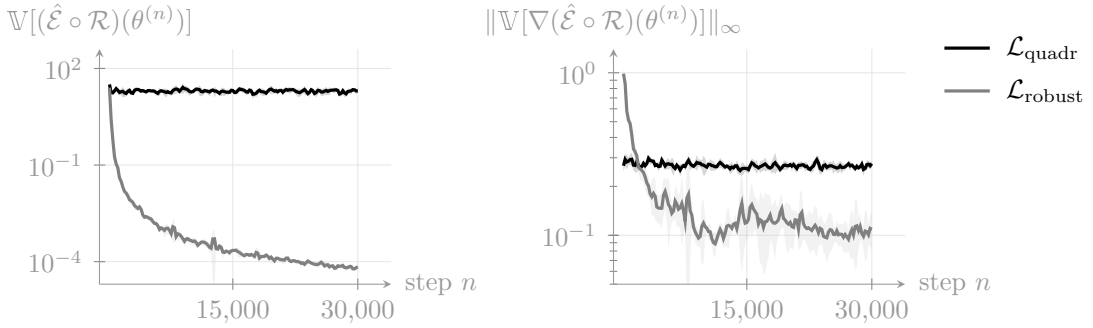


Figure 3.4: The variance of the empirical risk and the maximum variance of its gradient for the experiment in Figure 3.3. As predicted by our theory in Section 3.5, the variances are vanishing when converging to the minimum if one uses the loss function $\mathcal{L}_{\text{robust}}$. To simulate the stochastic integral occurring in $\mathcal{L}_{\text{robust}}$, we use the Euler-Maruyama method, see (2.15), with number of steps gradually increasing up to $K = 150$ at the last gradient step $N = 30000$. Compared to the quadratic loss function $\mathcal{L}_{\text{quadr}}$, this increases the average time per gradient step from $0.02s$ to $1.02s$. However, Figure 3.3 and Figure 3.7 below show that the loss function $\mathcal{L}_{\text{robust}}$ can also lead to significantly improved accuracies.

where the stochastic integral $\mathcal{I}_f$ can be thought of as a control variate. Since the definition of $\mathcal{I}_f$ in (3.18) depends on the same Brownian motion B as the SDE in (3.14), we define our data as

$$Z \coloneqq ((\xi, \tau, \Gamma), B)$$

such that both $Y = \varphi_\Gamma(S_T^{\xi,\tau,\Gamma})$ and $\mathcal{I}_f$ in (3.22) can be computed[3] from Z. Using (3.19), we see that the corresponding learning problem can be rewritten as

$$\mathbb{E}[\mathcal{L}_{\text{robust}}(f, Z)] = \mathbb{E}\left[(\bar{u}(X) - f(X))^2\right] + \mathbb{E}\left[\mathcal{I}_{\bar{u}-f}^2\right], \tag{3.23}$$

which can be compared to the decomposition of $\mathbb{E}[\mathcal{L}_{\text{quadr}}(f, Z)]$ in (2.3). The representation in (3.23) shows that the solution f^* to the learning problem with loss function $\mathcal{L}_{\text{robust}}$ is still given by the parametric solution map $\bar{u}$ of the Kolmogorov equation in (3.12). In addition, we note that the minimal risk for the loss function $\mathcal{L}_{\text{robust}}$ equals zero, as opposed to $\mathbb{V}[\mathcal{I}_{\bar{u}}]$ for the quadratic loss function $\mathcal{L}_{\text{quadr}}$, see (2.3) and (3.23). This can also be observed in Figure 3.3 and the experiments in Chapter V. More importantly, the identity in (3.19) implies that the variances

[3]To prove the existence of a mapping from the Brownian motion B to a corresponding SDE solution, we refer to Le Gall (2016, Theorem 8.5). Note that we could have used the same definition of the data Z also for the quadratic loss function in the previous sections.

of the empirical risk $\hat{\mathcal{E}}$, corresponding to the new loss function $\mathcal{L}_{\text{robust}}$, and its gradient vanish at the optimal solution $\mathcal{R}(\theta^*) = \bar{u}$, i.e.,

$$\mathbb{V}\left[(\hat{\mathcal{E}} \circ \mathcal{R})(\theta^*)\right] = \mathbb{V}\left[\nabla(\hat{\mathcal{E}} \circ \mathcal{R})(\theta^*)\right] = 0. \tag{3.24}$$

We further show that the variances in (3.24) are already reduced if

$$\nabla_x \mathcal{R}(\theta^{(N)}) \approx \nabla_x \bar{u}$$

in the sense of (3.5). This is supported by extensive numerical experiments in Chapter V, see also Figure 3.4 for an illustration. While the reduced variance significantly improves the accuracies in our numerical experiments, the stochastic integral $\mathcal{I}_f$ in the loss function $\mathcal{L}_{\text{robust}}$ needs to be discretized and requires pointwise evaluations of the gradient $\nabla_x f$. To find a trade-off between accuracy and complexity, the paper presents other variants of the loss function $\mathcal{L}_{\text{robust}}$, for instance, using a separate neural network to parametrize the gradient $\nabla_x f$. Overall, we can considerably improve the sample efficiency and accuracy of the method developed in Section 3.4, even for a fixed time or memory budget.

3.6 How Degenerate is the Parametrization of Neural Networks with the ReLU Activation Function?

The previous sections presented deep learning-based solvers for Kolmogorov equations and ways of obtaining bounds on the approximation and generalization errors. However, we implicitly assumed that we could find an empirical risk minimizer

$$\hat{f} \in \operatorname*{argmin}_{f \in \mathcal{H}} \hat{\mathcal{E}}(f),$$

see also the error decomposition in (3.7). It remains unclear why gradient descent should yield parameters $\theta^{(N)} \in \Theta$ for which the realization $\mathcal{R}(\theta^{(N)})$ has close to minimal empirical risk, i.e.,

$$\hat{\mathcal{E}}(\mathcal{R}(\theta^{(N)})) \approx \hat{\mathcal{E}}(\hat{f}).$$

In this paper, we consider a regularized hypothesis set of neural networks $\mathcal{H} \coloneqq \mathcal{R}(\Theta) \cap U$, where U is a convex and bounded subset of $\mathcal{M}$ w.r.t. to a suitable norm $\|\cdot\|$. For instance, U can be a ball in a Sobolev space known to contain the solution to the Kolmogorov equation in (2.9). We further assume that U can be well-approximated by corresponding regularized neural network parametrizations

$$\Theta^U \coloneqq \{\theta \in \Theta \colon \mathcal{R}(\theta) \in U\}$$

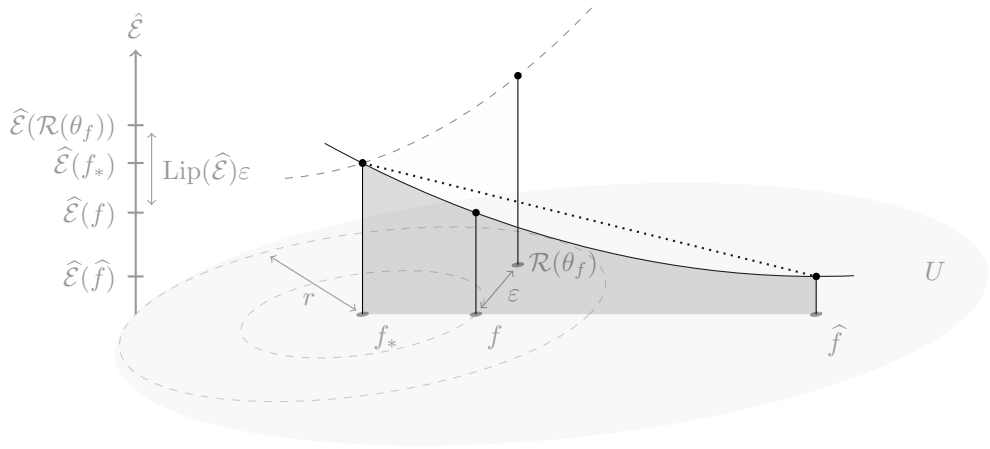


Figure 3.5: Illustration of the loss landscape around a local minimum f_* (with radius r) and a global minimum $\hat{f}$ in the (regularized) realization space $\mathcal{R}(\Theta^U) \subset U$, where the empirical risk $\hat{\mathcal{E}}$ is convex and Lipschitz continuous on U.

in the sense that

$$\sup_{f \in U} \inf_{\theta \in \Theta^U} \|\mathcal{R}(\theta) - f\| \leq \varepsilon. \tag{3.25}$$

For U consisting of functions with bounded $\mathcal{W}^{k,p}$-norm, we discussed estimates on the size of the neural network architecture to guarantee such approximation results in Section 3.2. Under natural[4] assumptions on the empirical risk $\hat{\mathcal{E}}$, namely that it is Lipschitz continuous and convex on U, this paper shows that every local minimum on the realization space $\mathcal{R}(\Theta^U)$ is either sharp or almost optimal. To see this, let $\hat{f}$ be a global and f_* be a local[5] minimum of

$$\min_{f \in \mathcal{R}(\Theta^U)} \hat{\mathcal{E}}(f). \tag{3.26}$$

Now define $\alpha \coloneqq \frac{r}{2\|\hat{f} - f_*\|}$ and

$$f \coloneqq (1-\alpha) f_* + \alpha \hat{f} \in U,$$

where $r \in (0, \infty)$ is the radius of the local minimum. Due to (3.25), there is $\theta_f \in \Theta^U$ such that $\|\mathcal{R}(\theta_f) - f\| \leq \varepsilon$ and, given that $r \geq 2\varepsilon$, it holds that

$$\hat{\mathcal{E}}(f_*) \leq \hat{\mathcal{E}}(\mathcal{R}(\theta_f)) \leq \hat{\mathcal{E}}(f) + \mathrm{Lip}(\hat{\mathcal{E}})\varepsilon \leq (1-\alpha)\hat{\mathcal{E}}(f_*) + \alpha \hat{\mathcal{E}}(\hat{f}) + \mathrm{Lip}(\hat{\mathcal{E}})\varepsilon,$$

[4]For instance, for a given realization of samples, the empirical risk w.r.t. the quadratic loss function $\mathcal{L}_{\mathrm{quadr}}$ in (2.2) is convex and locally Lipschitz continuous on $\mathcal{L}^\infty(\mathcal{X})$.

[5]We call $f_* \in F$ a local minimum of $\inf_{f \in F} E(f)$ of radius $r \in (0, \infty)$ if $E(f_*) \leq E(f)$ for all $f \in F$ with $\|f - f_*\| \leq r$.

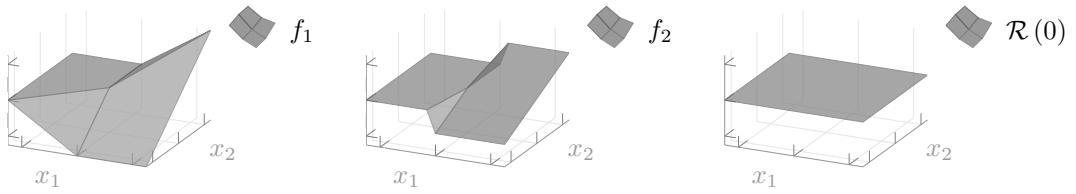


Figure 3.6: ReLU networks $f_k \colon [-1,1]^2 \to \mathbb{R}$, $x \mapsto k\varrho_R(kx_1) - k\varrho_R\left(kx_1 - \frac{1}{k^2}x_2\right)$, which convergence uniformly towards the zero function $\mathcal{R}(0)$ for $k \to \infty$. However, the norms of all parametrizations in $\mathcal{R}^{-1}(\{f_k\})$ with architecture $a = (2,2,1)$ tend towards infinity, see Chapter VI.

see also Figure 3.5. This implies that

$$\hat{\mathcal{E}}(f_*) - \hat{\mathcal{E}}(\hat{f}) \leq \frac{2\operatorname{diam}(U)\operatorname{Lip}(\hat{\mathcal{E}})\varepsilon}{r},$$

i.e., every local minimum f_* on the realization space $\mathcal{R}(\Theta^U)$ is either sharp ($r < 2\varepsilon$) or optimal up to $\mathcal{O}(\frac{\varepsilon}{r})$.

However, in practice, gradient descent is applied on the parameter space Θ and a local minimum of

$$\min_{\theta \in \Theta^U} \hat{\mathcal{E}}(\mathcal{R}(\theta)) \tag{3.27}$$

does not necessarily correspond to a local minimum of (3.26). For this to hold, proximity of neural network realizations needs to imply proximity of their parametrizations, i.e.,

$$\inf_{\theta \in \mathcal{R}^{-1}(\{f\})} \|\theta - \theta_*\|_\infty$$

can be bounded in terms of

$$\|f - \mathcal{R}(\theta_*)\|$$

for all $f \in \mathcal{R}(\Theta^U)$. However, for all nonlinear activation functions, this so-called *inverse stability* does not hold w.r.t. to the uniform norm $\|\cdot\| = \|\cdot\|_{\mathcal{L}^\infty}$, see Petersen, Raslan, and Voigtlaender (2020, Theorem 4.2) and Figure 3.6.

We characterize further degeneracies preventing inverse stability for ReLU networks, i.e., unbalanced magnitudes of the parameters in different layers and redundant neurons. Slightly enlarging a two-layer ReLU network and factoring out such pathologies, we end up with a parameter space $\hat{\Theta}$ such that inverse stability w.r.t. to the Sobolev $\mathcal{W}^{1,\infty}$-norm and $\mathcal{R}(\Theta) \subset \mathcal{R}(\hat{\Theta})$ holds. In other words, the hypothesis set $\mathcal{R}(\hat{\Theta})$ is at least as expressive as $\mathcal{R}(\Theta)$, but with certain degenerate parametrizations excluded so that the quality of the local minima in (3.27) can be estimated. In this sense, our results can also be viewed as principled ways of regularizing ReLU networks.

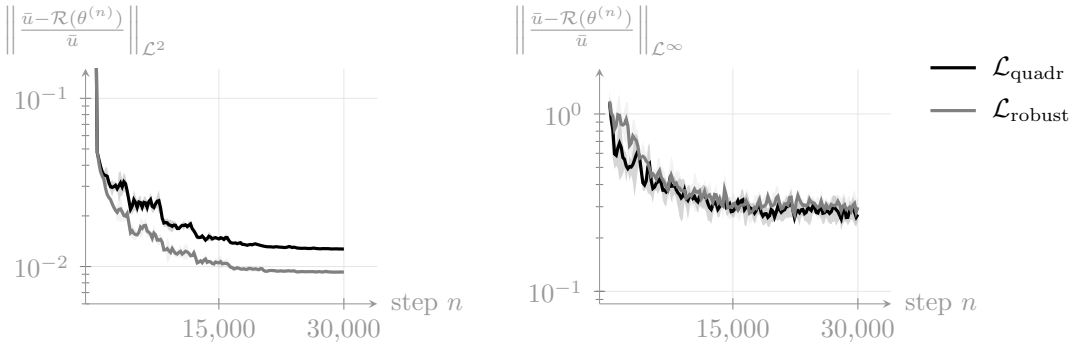


Figure 3.7: The relative $\mathcal{L}^2$ and $\mathcal{L}^\infty$-errors for the experiment in Figure 3.3. While the $\mathcal{L}^p$-norms (w.r.t. to the probability measure $\mathbb{P}_X$) are by definition increasing in p, there is a significant gap. As predicted by the learning theory in Section 3.3, the $\mathcal{L}^2$-errors are reasonably small, however, the $\mathcal{L}^\infty$-errors stagnate at a rather large value. Section 3.7 indicates that, without further prior knowledge, an intractable number of samples m might be required to reach a small error in the latter norm.

3.7 Learning ReLU Networks to High Uniform Accuracy is Intractable

Finally, we establish fundamental limits for learning ReLU networks from samples. While classical learning theory provides bounds on the error in the $\mathcal{L}^2$-norm, see also Section 3.3, we show that learning with uniform accuracy, i.e., measured in the $\mathcal{L}^\infty$-norm, might be intractable for realistic scenarios. Without further knowledge about the learning problem, we cannot guarantee high accuracies for all points in the input domain. This observation poses problems for safety-critical or scientific applications and sheds light on instabilities reported in the context of deep learning, such as *adversarial examples* (Szegedy et al., 2014). Moreover, our results provide theoretical evidence for the gap between the uniform accuracies guaranteed by approximation results, see Section 3.2, and the performance of deep learning in practice, as, for instance, observed by Adcock and Dexter (2021). In Figure 3.7, we see that also for our learning problem in the context of Kolmogorov equations, one can not always guarantee high uniform accuracies.

Let us sketch a simplified version of the result in the following. We assume that we only know a priori that[6]

$$f^* \in U \subset \mathcal{M} \coloneqq C([0,1]^d)$$

[6]For ease of presentation, we work on the domain $[0,1]^d$ and abbreviate $\mathcal{L}^\infty \coloneqq \mathcal{L}^\infty([0,1]^d)$. However, our results can readily be transferred to any bounded domain.

for some set U containing a sufficiently large class of ReLU networks. For instance, it is reasonable to assume that our hypothesis set $\mathcal{H} = \mathcal{R}(\Theta)$ of neural networks is contained in U. Specifically, we assume that there exists $u_0 \in U$ with

$$u_0 + \mathcal{R}(\{\theta \in \Theta \colon \|\theta\|_\infty \leq c\}) \subset U \tag{3.28}$$

for some bound $c \in (0, \infty)$ on the parameter magnitudes and an architecture $a = (d, 3d, \dots, 3d, 1) \in \mathbb{N}^{L+1}$ with $L \geq 3$ layers.

We want to compute the worst-case reconstruction error, i.e.,

$$\varepsilon \coloneqq \sup_{u \in U} \|\mathcal{A}(u) - u\|_{\mathcal{L}^\infty}, \tag{3.29}$$

for an *arbitrary* algorithm $\mathcal{A} \colon U \to \mathcal{L}^\infty$, which leverages m samples of the form $(X^{(i)}, u(X^{(i)}))_{i=1}^m$. In particular, we consider all variants of gradient descent and even potentially intractable algorithms, such as empirical risk minimization. We further emphasize that the samples do not need to be i.i.d. and can even be chosen in an adaptive fashion, as, e.g., done in *active learning*. In addition, we analyze the favorable case where the labels $(u(X^{(i)}))_{i=1}^m$ do not contain any noise.

We first note that the assumption in (3.28) implies that U contains functions of the form $u_0 \pm f_{y,M}$, where $f_{y,M} \in C([0,1]^d)$ are *bump functions* given by[7]

$$f_{y,M}(x) \coloneqq \frac{c^L (3d)^{L-2}}{2M} \varrho_R \left(\frac{M}{2} \sum_{i=1}^d h_{\frac{2}{M}} \left(x_i - y_i + \frac{1}{M} \right) - (d-1) \right), \tag{3.30}$$

for $x, y \in [0,1]^d$ and $M \in [1, \infty)$, as depicted in Figure 3.8. In the above, $h_{\frac{2}{M}}$ denotes the hat function defined in (3.2). To see that $u_0 \pm f_{y,M} \in U$, we can write $f_{y,M} = \mathcal{R}(\theta_{y,M})$, where the parameters $\theta_{y,M} = ((W^{(\ell)}, b^{(\ell)}))_{\ell=1}^L$ are given by[8]

$$\begin{aligned}
(W^{(1)}, b^{(1)}) &\coloneqq \left(c \left[\tfrac{\mathbf{1}_{d\times d}}{2}, \mathbf{1}_{d\times d}, \mathbf{0}_{d\times d} \right]^\top, c \left[\tfrac{M^{-1}\mathbf{1}_{1\times d} - y^\top}{2}, -y^\top, \tfrac{d-1}{2Md} \mathbf{1}_{1\times d} \right]^\top \right), \\
(W^{(2)}, b^{(2)}) &\coloneqq \left(c \left[\mathbf{1}_{3d\times d}, -\mathbf{1}_{3d\times d}, -\mathbf{1}_{3d\times d} \right], \mathbf{0}_{3d\times 1} \right), \\
(W^{(\ell)}, b^{(\ell)}) &\coloneqq (c\mathbf{1}_{3d\times 3d}, \mathbf{0}_{3d\times 1}), \quad \ell = 3, \dots, L-1, \quad \text{and} \\
(W^{(L)}, b^{(L)}) &\coloneqq (c\mathbf{1}_{1\times 3d}, 0).
\end{aligned}$$

We further note that the definition of the bump function in (3.30) implies that

$$\operatorname{supp}(f_{y,M}) \subset y + \left[\frac{1}{M}, \frac{1}{M} \right]^d \quad \text{and} \quad \|f_{y,M}\|_{\mathcal{L}^\infty} = \frac{c^L (3d)^{L-2}}{2M}. \tag{3.31}$$

[7] The functions $f_{y,M}$ are Lipschitz continuous and compactly supported, see (3.31). Different from other notions of bump functions, they are, however, not smooth, i.e., in $C^\infty([0,1]^d)$. We also note that their construction is similar to the approximate characteristic function in (3.6).

[8] For $k, d \in \mathbb{N}$, we denote by $\mathbf{1}_{k,d} \in \mathbb{R}^{k\times d}$ and $\mathbf{0}_{k,d} \in \mathbb{R}^{k\times d}$ the matrices containing only ones and zeros, respectively.

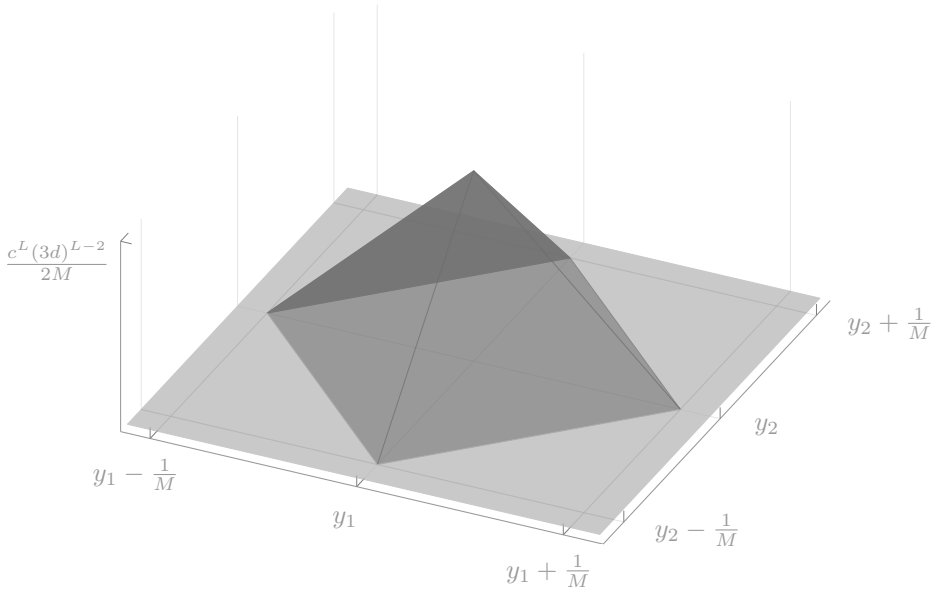


Figure 3.8: The bump function $f_{y,M}$ in (3.30) for $d = 2$, which can be represented by a ReLU network $\mathcal{R}(\theta_{y,M})$ with architecture $a = (d, 3d, \dots, 3d, 1) \in \mathbb{N}^{L+1}$ and parameters $\theta_{y,M}$ bounded by $c \in (0, \infty)$.

Now let $\big((X^{(i)}, u_0(X^{(i)})\big)_{i=1}^m$ be the samples used by a given algorithm $\mathcal{A}$ for reconstructing the function u_0 in (3.28). Due to (3.31), we can choose $M := 4\lceil m^{1/d} \rceil$ and consider a grid of width $\frac{1}{2M}$ to find $y \in \left[\frac{1}{4M}, 1 - \frac{1}{4M}\right]^d$ such that

$$f_{y,M}(X^{(i)}) = 0, \quad i = 1, \dots, m.$$

As $\mathcal{A}$ only depends on the function values at the features $(X^{(i)})_{i=1}^m$, we have that $\mathcal{A}(u_0) = \mathcal{A}(u_0 \pm f_{y,M})$. Together with (3.31) and the triangle inequality, this establishes that

$$\begin{aligned}
\varepsilon &= \sup_{u \in U} \|\mathcal{A}(u) - u\|_{\mathcal{L}^\infty} \\
&\geq \frac{1}{2}\|\mathcal{A}(u_0 + f_{y,M}) - (u_0 + f_{y,M})\|_{\mathcal{L}^\infty} + \frac{1}{2}\|\mathcal{A}(u_0 - f_{y,M}) - (u_0 - f_{y,M})\|_{\mathcal{L}^\infty} \\
&\geq \frac{1}{2}\|\mathcal{A}(u_0) - (u_0 + f_{y,M})\|_{\mathcal{L}^\infty} + \frac{1}{2}\|\mathcal{A}(u_0) - (u_0 - f_{y,M})\|_{\mathcal{L}^\infty} \\
&\geq \|f_{y,M})\|_{\mathcal{L}^\infty} = \frac{c^L(3d)^{L-2}}{2M} \geq \frac{c^L(3d)^{L-2}}{16m^{1/d}}.
\end{aligned}$$

As our choice of the algorithm $\mathcal{A}$ was arbitrary, we have shown that any algorithm that only depends on point evaluations requires at least

$$m \geq \frac{1}{16} c^{dL} (3d)^{d(L-2)} \varepsilon^{-d}$$

samples to achieve a uniform accuracy of ε for all functions in U, as defined in (3.29). In particular, the number of required samples depends *exponentially* on the input dimension d and the depth L of the neural networks, which quickly renders the problem intractable for realistic scenarios. While the expressivity of ReLU networks leads to efficient approximation results in the $\mathcal{L}^\infty$-norm, see Section 3.7, it also allows realizing sharp bump functions $f_{y,M}$ with relatively few bounded parameters. The number of samples m required to learn the representations in such approximation results can therefore massively exceed the number of parameters $\dim(\Theta)$ defining the hypothesis set. We note that other hypothesis sets, e.g., subsets of polynomials or certain reproducing kernel Hilbert spaces, show different behavior and can realize their approximation rates from samples.

The paper further considers more general *randomized* algorithms $\mathcal{A}$ and provides similar estimates for $\mathcal{L}^p$-reconstruction errors subject to ℓ^q-regularization on the neural network parameters, where $p, q \in [1, \infty]$. We also construct an asymptotically sharp upper bound based on piecewise interpolation of Lipschitz continuous functions and corroborate the results with numerical experiments.

4 Discussion

Let us recall the central question posed in the introduction regarding the theoretical performance of deep learning and, in particular, the analysis of the errors in (2.16) for numerically solving Kolmogorov equations. In the previous chapters, we formalized this question and provided corresponding theoretical guarantees, often by focusing on sub-problems concerning the *approximation*, *generalization*, and *optimization errors*. Let us summarize our contributions to each of these problems in the following:

1. **Approximation error:** *How well does the best approximation*

$$f_{\mathcal{H}}^* \in \operatorname*{argmin}_{f \in \mathcal{H}} \mathcal{E}(f)$$

in a hypothesis set $\mathcal{H} = \mathcal{R}(\Theta)$ of neural networks approximate the solution

$$f^* \in \operatorname*{argmin}_{f \in \mathcal{M}} \mathcal{E}(f)$$

to the learning problem?

If we know that f^* obeys certain regularity properties, such as smoothness, compositionality, or symmetries, there are a number of results on the required size of the neural network architecture needed to approximate f^* up to a desired precision, see Chapter I. For f^* belonging to certain Sobolev spaces, the simultaneous approximation of f^* and its derivative has been outlined in Section 3.2. Such approximation results are often obtained via local polynomial approximation, which, however, suffers from the curse of dimensionality. In specific scenarios, one can achieve approximation results where the size of the neural networks scales only polynomially in the underlying dimension, see Section 3.1 for an overview. For instance, we have seen how we can frame the solution of parametric families of Kolmogorov equations as a learning problem and how neural networks can approximate the solution f^* without the curse of dimensionality by emulating uniform Monte Carlo approximations, see Sections 3.3 and 3.4.

2. **Generalization error:** *How well does the empirical risk minimizer*

$$\hat{f} \in \operatorname*{argmin}_{f \in \mathcal{H}} \hat{\mathcal{E}}(f)$$

generalize, i.e., minimize the risk $\mathcal{E}$?

One can compute certain intrinsic dimensions of the hypothesis set $\mathcal{H}$, which, together with concentration inequalities, can be used to bound the number of samples m needed such that the risk of the empirical risk minimizer is almost optimal, i.e., $\mathcal{E}(\hat{f}) \approx \mathcal{E}(f_{\mathcal{H}}^*)$, see Chapter I. For instance, using the Lipschitz continuity of $\mathcal{R}$ and covering numbers of the parameter space Θ, one can compute covering numbers of neural network hypothesis sets $\mathcal{H} = \mathcal{R}(\Theta)$, see Section 3.3. If f^* can be approximated without the curse of dimensionality, such bounds can be used to show that also the number of samples required to bound the generalization error, i.e., $\mathcal{E}(\hat{f}) - \mathcal{E}(f_{\mathcal{H}}^*) \leq \varepsilon$, scales only polynomially in the underlying dimension d and the reciprocal accuracy ε^{-1}, see Section 3.3. Based on the approximation results established for Kolmogorov equations, this implies that empirical risk minimization over deep neural networks can overcome the curse of dimensionality in the numerical solution of Kolmogorov equations.

3. **Optimization error:** *How well does (stochastic) gradient descent, applied to the mapping*

$$\theta \mapsto (\hat{\mathcal{E}} \circ \mathcal{R})(\theta), \tag{4.1}$$

minimize the empirical risk $\hat{\mathcal{E}}$?

Several works show that the mapping in (4.1) can be well-behaved for the use of gradient descent despite its non-convexity, see Chapter I. For instance, reasonably large hypothesis sets of neural networks can be viewed as almost convex, given that they can approximate well certain convex sets of functions $U \subset \mathcal{M}$. In particular, this guarantees that all sufficiently wide local minima of the empirical risk minimization problem over the corresponding regularized hypothesis set, i.e., $\min_{f \in \mathcal{R}(\Theta) \cap U} \hat{\mathcal{E}}(f)$, are nearly optimal, see Section 3.6. However, to estimate the quality of local minima for the practically relevant minimization problem on the parameter space, i.e., $\min_{\theta \in \Theta \cap \mathcal{R}^{-1}(U)} (\hat{\mathcal{E}} \circ \mathcal{R})(\theta)$, one needs to factor out a series of degeneracies from the parametrizations. For two-layer ReLU networks, we have successfully characterized these degeneracies in Chapter VI. Empirically, we found that neural networks are capable of numerically solving high-dimensional (parametric families of) Kolmogorov equations that are completely out of reach for classical methods, see Section 3.4. In Section 3.5, we developed a loss function that further improves performance by reducing the variance of the gradients of the mapping in (4.1). However, we also noticed that it is generally hard to achieve high uniform accuracies with neural networks. In Section 3.7, we showed that a hypothesis set of ReLU networks $\mathcal{R}(\Theta) \subset \mathcal{M}$ can contain sharp bump functions, such that, without further knowledge about $f^* \in \mathcal{M}$, an intractable number of samples might be needed to identify f^* with high uniform accuracy.

Most of the aforementioned results make use of assumptions on the learning problem which are hard to verify in practice, such as regularity properties of f^* or i.i.d. assumptions on the samples $(Z^{(i)})_{i=1}^m$. However, we showed how scientific machine learning tasks, for instance, arising in the context of PDEs, provide suitable settings, where such assumptions can be guaranteed. This leads to rigorous results on the theoretical performance of corresponding algorithms, which is a rare scenario for applications of deep learning, often regarded as black-box algorithms.

However, we also point out that there are still a few shortcomings in our analysis. For the results on the optimality of local minima to be applicable in practical settings, e.g., with subsets U of certain smoothness spaces, we need to investigate the degeneracies of *deep* networks with smoother activation functions in order to guarantee inverse stability. As outlined in Section 3.7, there is also a gap between theoretically achievable approximation rates and the rates obtained in practice. One can observe that the neural network parametrizations constructed in such approximation results are often not robust w.r.t. perturbations and might thus correspond to sharp minima in the loss landscape of the empirical risk minimization problem. While our bounds are useful in an asymptotic sense, proving the absence of the curse of dimensionality, they do not provide us with practical guidance on how to choose the number of samples and the architectures of the neural networks for a specific task.

As results from classical learning theory typically bound the generalization error uniformly over the whole hypothesis set $\mathcal{H}$, such bounds can also be completely vacuous for certain real-world applications. In Section 3.1, we observed that the performance of deep learning often depends on an intricate interplay between the chosen neural network architecture and the optimization scheme, questioning whether a separate analysis of approximation, generalization, and optimization errors can lead to tight bounds. Nevertheless, we believe that our results provide a solid starting point for a rigorous mathematical analysis of deep learning algorithms. While we used Kolmogorov equations and their SDE-based representations as a convenient exemplary application, many of the results in this thesis can be applied to other settings and applications as well.

In this context, we want to mention a more general way of transforming the numerical solution of PDEs into a learning problem, called *Physics Informed Neural Networks* (PINNs), see Raissi, Perdikaris, and Karniadakis (2019). Instead of leveraging SDE-based representations, PINNs directly minimize the pointwise residual of the PDE. For instance, for the Kolmogorov equation in (2.9), one can use the loss function

$$\mathcal{L}(f, Z) := |(\partial_t f + Ff)(\xi, \tau)|^2 + c|f(\bar{\xi}, T) - \varphi(\bar{\xi})|^2, \quad f \in \mathcal{M}, \tag{4.2}$$

where $c \in (0, \infty)$ is a penalty parameter, $Z = (\xi, \bar{\xi}, \tau)$ is a suitable random variable distributed on $D \times D \times [0, T]$, and $\mathcal{M} \subset C^{2,1}(\mathbb{R}^d \times [0, T])$ is a set of sufficiently regular functions. When using a hypothesis set of neural networks $\mathcal{R}(\Theta) \subset \mathcal{M}$, the derivatives appearing in the differential operator F in (4.2) can, for instance, be computed using automatic differentiation.

While the PINN framework can readily be applied to a wide range of PDEs, it can be sensitive to the choice of penalty parameters, such as c in (4.2), and potentially unstable, as discussed by Krishnapriyan et al. (2021) and S. Wang, Teng, and Perdikaris (2021). In addition, computing the loss in (4.2) requires the evaluation of pointwise derivatives, which can be computationally expensive. For special classes of PDEs, one can leverage SDE-based representations, as in the case of Kolmogorov equations, or weak formulations of PDEs to mitigate these issues. The work by Nüsken and Richter (2021a) shows that one can also interpolate between SDE-based formulations and PINNs, providing a unified view of these approaches.

For the solution of parametric PDEs, PINNs have been combined with *neural operators* (Li et al., 2021; S. Wang and Perdikaris, 2023). The latter architectures learn a discretization-invariant approximation to the operator mapping PDE coefficient functions and boundary conditions, such as σ, μ, and φ, to the corresponding solution u. We note that analyses of the approximation and generalization errors, similar to the ones in this thesis, have since then been carried out for PINNs (De Ryck and Mishra, 2022), neural operators (Lanthaler, Mishra, and Karniadakis, 2022), weak formulations of elliptic PDEs (Y. Lu, J. Lu, and M. Wang, 2021), and general neural network training (Beck, Jentzen, and Kuckuck, 2022).

We also want to highlight potential extensions of our work within the context of Kolmogorov equations. First, note that there are also SDE-based representations for nonlinear problems based on *backward SDEs* (Beck, E, and Jentzen, 2019; Han, Jentzen, and E, 2018) and for bounded domains and elliptic PDEs using *stopping times* for the SDE solutions, see, e.g., Baldi (2017). Starting from the theoretical analysis presented by Grohs and Herrmann (2022), a promising direction is to leverage *walk-on-spheres methods* for the solution of elliptic PDEs on complex domains. Another interesting extension would be the combination of SDE-based formulations with neural operators.

Finally, we want to mention the importance of the PDEs considered in this work in *optimal control* and *generative modeling*. Specifically, solutions to Hamilton-Jacobi-Bellman equations, given as Hopf-Cole transforms of Kolmogorov equations, are intimately connected to optimal control and sampling problems (Dai Pra, 1991; Fleming and Soner, 2006; Nüsken and Richter, 2021b; Pavon, 1989; Tzen and Raginsky, 2019). Moreover, so-called *diffusion models*, which have established

themselves as state-of-the-art in generative modeling of high-dimensional image data, can be analyzed using *reverse-time SDEs*, the density of which is governed by Kolmogorov equations (Berner, Richter, and Ullrich, 2022; Huang, Lim, and Courville, 2021). These connections allow transferring both the theoretical results and the practical PDE solvers presented in this thesis to a variety of applications.

In conclusion, the present thesis provides a solid foundation for future research in the field of mathematical analysis of deep learning. Special emphasis is placed on the study and development of methods that reside at the interface of deep learning and differential equations. As such, this effort represents a step toward a rigorous and comprehensive analysis of deep learning for scientific problems, ultimately culminating in well-founded algorithms that advance the natural sciences.

References

Adcock, Ben and Nick Dexter (2021). "The gap between theory and practice in function approximation with deep neural networks." In: *SIAM Journal on Mathematics of Data Science* 3.2, pp. 624–655. DOI: 10.1137/20M131309X.

Baldi, Paolo (2017). *Stochastic Calculus: An Introduction Through Theory and Exercises*. Universitext. Springer International Publishing. DOI: 10.1007/978-3-319-62226-2.

Beck, Christian, Sebastian Becker, Philipp Grohs, Nor Jaafari, and Arnulf Jentzen (2021). "Solving the Kolmogorov PDE by means of deep learning." In: *Journal of Scientific Computing* 88.3, pp. 1–28. DOI: 10.1007/s10915-021-01590-0.

Beck, Christian, Weinan E, and Arnulf Jentzen (2019). "Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations." In: *Journal of Nonlinear Science* 29.4, pp. 1563–1619. DOI: 10.1007/s00332-018-9525-3.

Beck, Christian, Arnulf Jentzen, and Benno Kuckuck (2022). "Full error analysis for the training of deep neural networks." In: *Infinite Dimensional Analysis, Quantum Probability and Related Topics* 25.02, p. 2150020. DOI: 10.1142/S021902572150020X.

Berner, Julius (2018). "Solving stochastic differential equations and Kolmogorov equations by means of deep learning and Multilevel Monte Carlo simulation." MA thesis. University of Vienna.

Berner, Julius, Markus Dablander, and Philipp Grohs (2020). "Numerically Solving Parametric Families of High-Dimensional Kolmogorov Partial Differential Equations via Deep Learning." In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 16615–16627.

Berner, Julius, Dennis Elbrächter, and Philipp Grohs (2019). "How degenerate is the parametrization of neural networks with the ReLU activation function?" In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., pp. 7790–7801.

Berner, Julius, Dennis Elbrächter, Philipp Grohs, and Arnulf Jentzen (2019). "Towards a regularity theory for ReLU networks–chain rule and global error esti-

mates." In: *2019 13th International conference on Sampling Theory and Applications (SampTA)*. IEEE, pp. 1–5. DOI: 10.1109/SampTA45681.2019.9031005.

Berner, Julius, Philipp Grohs, and Arnulf Jentzen (2020). "Analysis of the Generalization Error: Empirical Risk Minimization over Deep Artificial Neural Networks Overcomes the Curse of Dimensionality in the Numerical Approximation of Black–Scholes Partial Differential Equations." In: *SIAM Journal on Mathematics of Data Science* 2.3, pp. 631–657. DOI: 10.1109/IWOBI.2017.7985525.

Berner, Julius, Philipp Grohs, Gitta Kutyniok, and Philipp Petersen (2022). "The Modern Mathematics of Deep Learning." In: *Mathematical Aspects of Deep Learning*. Cambridge University Press, pp. 1–111. DOI: 10.1017/9781009025096.002.

Berner, Julius, Philipp Grohs, and Felix Voigtlaender (2023). "Learning ReLU networks to high uniform accuracy is intractable." In: *International Conference on Learning Representations*.

Berner, Julius, Lorenz Richter, and Karen Ullrich (2022). "An optimal control perspective on diffusion-based generative modeling." In: *NeurIPS 2022 Workshop on Score-Based Methods*.

Dai Pra, Paolo (1991). "A stochastic control approach to reciprocal diffusion processes." In: *Applied mathematics and Optimization* 23.1, pp. 313–329. DOI: 10.1007/BF01442404.

De Ryck, Tim and Siddhartha Mishra (2022). "Error analysis for physics-informed neural networks (PINNs) approximating Kolmogorov PDEs." In: *Advances in Computational Mathematics* 48.6, pp. 1–40. DOI: 10.1007/s10444-022-09985-9.

Elbrächter, Dennis, Dmytro Perekrestenko, Philipp Grohs, and Helmut Bölcskei (2021). "Deep neural network approximation theory." In: *IEEE Transactions on Information Theory* 67.5, pp. 2581–2623. DOI: 10.1109/TIT.2021.3062161.

Evans, Lawrence C (2010). *Partial differential equations*. Vol. 19. American Mathematical Society. DOI: 10.1090/gsm/019.

Fleming, Wendell H and Halil Mete Soner (2006). *Controlled Markov processes and viscosity solutions*. Vol. 25. Springer Science & Business Media. DOI: 10.1007/0-387-31071-1.

Géron, Aurelien (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.

Giles, Michael B (2008). "Multilevel Monte Carlo path simulation." In: *Operations research* 56.3, pp. 607–617. DOI: 10.1287/opre.1070.0496.

Grohs, Philipp and Lukas Herrmann (2022). "Deep neural network approximation for high-dimensional elliptic PDEs with boundary conditions." In: *IMA Journal of Numerical Analysis* 42.3, pp. 2055–2082. DOI: 10.1093/imanum/drab031.

Grohs, Philipp, Fabian Hornung, Arnulf Jentzen, and Philippe Von Wurstemberger (2020). "A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations." In: *Memoirs of the American Mathematical Society*.

Gühring, Ingo, Gitta Kutyniok, and Philipp Petersen (2020). "Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms." In: *Analysis and Applications* 18.05, pp. 803–859. DOI: 10.1142/S0219530519410021.

Hairer, Martin, Martin Hutzenthaler, and Arnulf Jentzen (2015). "Loss of regularity for Kolmogorov equations." In: *The Annals of Probability* 43.2, pp. 468–527. DOI: 10.1214/13-AOP838.

Han, Jiequn, Arnulf Jentzen, and Weinan E (2018). "Solving high-dimensional partial differential equations using deep learning." In: *Proceedings of the National Academy of Sciences* 115.34, pp. 8505–8510. DOI: 10.1073/pnas.1718942115.

Huang, Chin-Wei, Jae Hyun Lim, and Aaron C Courville (2021). "A variational perspective on diffusion-based generative models and score matching." In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., pp. 22863–22876.

Jentzen, Arnulf, Diyora Salimova, and Timo Welti (2021). "A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients." In: *Communications in Mathematical Sciences* 19.5, pp. 1167–1205. DOI: 10.4310/CMS.2021.v19.n5.a1.

Kloeden, Peter E and Eckhard Platen (1992). *Numerical Solution of Stochastic Differential Equations*. Stochastic Modelling and Applied Probability. Springer. DOI: 10.1007/978-3-662-12616-5.

Krishnapriyan, Aditi, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney (2021). "Characterizing possible failure modes in physics-informed neural networks." In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., pp. 1097–1105.

Lanthaler, Samuel, Siddhartha Mishra, and George E Karniadakis (2022). "Error estimates for DeepONets: a deep learning framework in infinite dimensions." In: *Transactions of Mathematics and Its Applications* 6.1. DOI: 10.1093/imatrm/tnac001.

Le Gall, Jean-François (2016). *Brownian motion, martingales, and stochastic calculus*. Springer. DOI: https://doi.org/10.1007/978-3-319-31089-3.

Li, Zongyi, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar (2021). *Physics-informed neural operator for learning partial differential equations*. arXiv: 2111.03794.

Lu, Yulong, Jianfeng Lu, and Min Wang (2021). "A priori generalization analysis of the deep Ritz method for solving high dimensional elliptic partial differential equations." In: *Proceedings of Thirty Fourth Conference on Learning Theory*. PMLR, pp. 3196–3241.

McCulloch, Warren S and Walter Pitts (1943). "A logical calculus of the ideas immanent in nervous activity." In: *The Bulletin of Mathematical Biophysics* 5.4, pp. 115–133. DOI: 10.1007/BF02478259.

Nüsken, Nikolas and Lorenz Richter (2021a). *Interpolating between BSDEs and PINNs–deep learning for elliptic and parabolic boundary value problems*. arXiv: 2112.03749.

– (2021b). "Solving high-dimensional Hamilton–Jacobi–Bellman PDEs using neural networks: perspectives from the theory of controlled diffusions and measures on path space." In: *Partial Differential Equations and Applications* 2.4, pp. 1–48. DOI: 10.1007/s42985-021-00102-x.

Pavon, Michele (1989). "Stochastic control and nonequilibrium thermodynamical systems." In: *Applied Mathematics and Optimization* 19.1, pp. 187–202. DOI: 10.1007/BF01448198.

Petersen, Philipp, Mones Raslan, and Felix Voigtlaender (2020). "Topological properties of the set of functions generated by neural networks of fixed size." In: *Foundations of Computational Mathematics*, pp. 1–70. DOI: 10.1007/s10208-020-09461-0.

Raissi, Maziar, Paris Perdikaris, and George E Karniadakis (2019). "Physics-informed neural networks: A deep learning framework for solving forward and

inverse problems involving nonlinear partial differential equations." In: *Journal of Computational Physics* 378, pp. 686–707. DOI: 10.1016/j.jcp.2018.10.045.

Reisinger, Christoph and Yufei Zhang (2020). "Rectified deep neural networks overcome the curse of dimensionality for nonsmooth value functions in zero-sum games of nonlinear stiff systems." In: *Analysis and Applications* 18.06, pp. 951–999. DOI: 10.1142/S0219530520500116.

Richter, Lorenz and Julius Berner (2022). "Robust SDE-Based Variational Formulations for Solving Linear PDEs via Deep Learning." In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 18649–18666.

Rosenblatt, Frank (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386. DOI: 10.1037/h0042519.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors." In: *Nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0.

Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus (2014). "Intriguing properties of neural networks." In: *International Conference on Learning Representations*.

Tzen, Belinda and Maxim Raginsky (2019). "Theoretical guarantees for sampling and inference in generative models with latent diffusions." In: *Proceedings of the Thirty-Second Conference on Learning Theory*. PMLR, pp. 3084–3114.

Wang, Sifan and Paris Perdikaris (2023). "Long-time integration of parametric evolution equations with physics-informed DeepONets." In: *Journal of Computational Physics* 475, p. 111855. DOI: 10.1016/j.jcp.2022.111855.

Wang, Sifan, Yujun Teng, and Paris Perdikaris (2021). "Understanding and mitigating gradient flow pathologies in physics-informed neural networks." In: *SIAM Journal on Scientific Computing* 43.5, A3055–A3081. DOI: 10.1137/20M1318043.

Yarotsky, Dmitry (2017). "Error bounds for approximations with deep ReLU networks." In: *Neural Networks* 94, pp. 103–114. DOI: https://doi.org/10.1016/j.neunet.2017.07.002.

Publications

I The Modern Mathematics of Deep Learning

Comments

E-Print: arXiv:2105.04026[cs.LG]

Contribution

Gitta Kutyniok and Philipp Grohs provided drafts for Chapters 7 and 8, respectively. The remaining content and the final version were conceived, developed, and written in equal parts by Philipp Petersen and Julius Berner.

Bibliographic Information

Berner, Julius, Philipp Grohs, Gitta Kutyniok, and Philipp Petersen (2022). "The Modern Mathematics of Deep Learning." In: *Mathematical Aspects of Deep Learning*. Cambridge University Press, pp. 1–111. DOI: 10.1017/9781009025096.002.

Copyright Notice

The Modern Mathematics of Deep Learning*

Julius Berner† Philipp Grohs‡ Gitta Kutyniok§ Philipp Petersen‡

Abstract

We describe the new field of mathematical analysis of deep learning. This field emerged around a list of research questions that were not answered within the classical framework of learning theory. These questions concern: the outstanding generalization power of overparametrized neural networks, the role of depth in deep architectures, the apparent absence of the curse of dimensionality, the surprisingly successful optimization performance despite the non-convexity of the problem, understanding what features are learned, why deep architectures perform exceptionally well in physical problems, and which fine aspects of an architecture affect the behavior of a learning task in which way. We present an overview of modern approaches that yield partial answers to these questions. For selected approaches, we describe the main ideas in more detail.

Contents

*A version of this review paper appears as a chapter in the book "Mathematical Aspects of Deep Learning" by Cambridge University Press.

†Faculty of Mathematics, University of Vienna.

‡Faculty of Mathematics and Research Network DataScience@UniVienna, University of Vienna.

§Department of Mathematics, Ludwig Maximilian University of Munich, and Department of Physics and Technology, University of Tromsø.

1 Introduction

Deep learning has undoubtedly established itself as the outstanding machine learning technique of recent times. This dominant position was claimed through a series of overwhelming successes in widely different application areas.

Perhaps the most famous application of deep learning and certainly one of the first where these techniques became state-of-the-art is image classification [LBBH98, KSH12, SLJ+15, HZRS16]. In this area, deep learning is nowadays the only method that is seriously considered. The prowess of deep learning classifiers goes so far that they often outperform humans in image labelling tasks [HZRS15].

A second famous application area is the training of deep-learning-based agents to play board games or computer games, such as Atari games [MKS+13]. In this context, probably the most prominent achievement yet is the development of an algorithm that beat the best human player in the game of Go [SHM+16, SSS+17]—a feat that was previously unthinkable owing to the extreme complexity of this game. Besides, even in multiplayer, team-based games with incomplete information deep-learning-based agents nowadays outperform world-class human teams [BBC+19, VBC+19].

In addition to playing games, deep learning has also led to impressive breakthroughs in the natural sciences. For example, it is used in the development of drugs [MSL+15], molecular dynamics [FHH+17], or in high-energy physics [BSW14]. One of the most astounding recent breakthroughs in scientific applications is the development of a deep-learning-based predictor for the folding behavior of proteins [SEJ+20]. This predictor is the first method to match the accuracy of lab-based methods.

Finally, in the vast field of natural language processing, which includes the subtasks of understanding, summarizing, or generating text, impressive advances were made based on deep learning. Here, we refer to [YHPC18] for an overview. One technique that recently stood out is based on a so-called transformer neural network [BCB15, VSP+17]. This network structure gave rise to the impressive GPT-3 model [BMR+20] which not only creates coherent and compelling texts but can also produce code, such as, for the layout of a webpage according to some instructions that a user inputs in plain English. Transformer neural networks have also been successfully employed in the field of symbolic mathematics [SGHK18, LC19].

In this article, we present and discuss the mathematical foundations of the success story outlined above. More precisely, our goal is to outline the newly emerging field of *mathematical analysis of deep learning*. To accurately describe this field, a necessary preparatory step is to sharpen our definition of the term deep learning. For the purposes of this article, we will use the term in the following narrow sense: *Deep learning refers to techniques where deep neural networks*[1] *are trained with gradient-based methods.* This narrow

[1] We will define the term *neural network* later but, for this definition, one can view it as a parametrized family of functions with a differentiable parametrization.

definition is helpful to make this article more concise. We would like to stress, however, that we do not claim in any way that this is the *best* or the *right* definition of deep learning.

Having fixed a definition of deep learning, three questions arise concerning the aforementioned emerging field of mathematical analysis of deep learning: To what extent is a mathematical theory necessary? Is it truly a new field? What are the questions studied in this area?

Let us start by explaining the necessity of a theoretical analysis of the tools described above. From a scientific perspective, the primary reason why deep learning should be studied mathematically is simple curiosity. As we will see throughout this article, many practically observed phenomena in this context are not explained theoretically. Moreover, theoretical insights and the development of a comprehensive theory are often the driving force underlying the development of new and improved methods. Prominent examples of mathematical theories with such an effect are the theory of fluid mechanics which is an invaluable asset to the design of aircraft or cars and the theory of information that affects and shapes all modern digital communication. In the words of Vladimir Vapnik[2]: "Nothing is more practical than a good theory", [Vap13, Preface]. In addition to being interesting and practical, theoretical insight may also be necessary. Indeed, in many applications of machine learning, such as medical diagnosis, self-driving cars, and robotics, a significant level of control and predictability of deep learning methods is mandatory. Also, in services, such as banking or insurance, the technology should be controllable to guarantee fair and explainable decisions.

Let us next address the claim that the field of mathematical analysis of deep learning is a newly emerging area. In fact, under the aforementioned definition of deep learning, there are two main ingredients of the technology: deep neural networks and gradient-based optimization. The first artificial neuron was already introduced in 1943 in [MP43]. This neuron was not trained but instead used to explain a biological neuron. The first multi-layered network of such artificial neurons that was also trained can be found in [Ros58]. Since then, various neural network architectures have been developed. We will discuss these architectures in detail in the following sections. The second ingredient, gradient-based optimization, is made possible by the observation that due to the graph-based structure of neural networks the gradient of an objective function with respect to the parameters of the neural network can be computed efficiently. This has been observed in various ways, see [Kel60, Dre62, Lin70, RHW86]. Again, these techniques will be discussed in the upcoming sections. Since then, techniques have been improved and extended. As the rest of the manuscript is spent reviewing these methods, we will keep the discussion of literature at this point brief. Instead, we refer to some overviews of the history of deep learning from various perspectives: [LBH15, Sch15, GBC16, HH19].

Given the fact that the two main ingredients of deep neural networks have been around for a long time, one would expect that a comprehensive mathematical theory has been developed that describes why and when deep-learning-based methods will perform well or when they will fail. Statistical learning theory [AB99, Vap99, CS02, BBL03, Vap13] describes multiple aspects of the performance of general learning methods and in particular deep learning. We will review this theory in the context of deep learning in Subsection 1.2 below. Hereby, we focus on classical, deep learning-related results that we consider well-known in the machine learning community. Nonetheless, the choice of these results is guaranteed to be subjective. We will find that the presented, classical theory is too general to explain the performance of deep learning adequately. In this context, we will identify the following questions that appear to be difficult to answer within the classical framework of learning theory: *Why do trained deep neural networks not overfit on the training data despite the enormous power of the architecture? What is the advantage of deep compared to shallow architectures? Why do these methods seemingly not suffer from the curse of dimensionality? Why does the optimization routine often succeed in finding good solutions despite the non-convexity, non-linearity, and often non-smoothness of the problem? Which aspects of an architecture affect the performance of the associated models and how? Which features of data are learned by deep architectures? Why do these methods perform as well as or better than specialized numerical tools in natural sciences?*

The new field of mathematical analysis of deep learning has emerged around questions like the ones listed above. In the remainder of this article, we will collect some of the main recent advances to answer these questions. Because this field of mathematical analysis of deep learning is incredibly active and new material is added at breathtaking speeds, a brief survey on recent advances in this area is guaranteed to miss not only

[2]This claim can be found earlier in a non-mathematical context in the works of Kurt Lewin [Lew43].

a couple of references but also many of the most essential ones. Therefore, we do not strive for a complete overview, but instead, showcase several fundamental ideas on a mostly intuitive level. In this way, we hope to allow the reader to familiarize themselves with some exciting concepts and provide a convenient entry-point for further studies.

1.1 Notation

We denote by $\mathbb{N}$ the set of natural numbers, by $\mathbb{Z}$ the set of integers and by $\mathbb{R}$ the field of real numbers. For $N \in \mathbb{N}$, we denote by $[N]$ the set $\{1, \dots, N\}$. For two functions $f, g \colon \mathcal{X} \to [0, \infty)$, we write $f \lesssim g$, if there exists a universal constant c such that $f(x) \leq cg(x)$ for all $x \in \mathcal{X}$. In a pseudometric space $(\mathcal{X}, d_{\mathcal{X}})$, we define the ball of radius $r \in (0, \infty)$ around a point $x \in \mathcal{X}$ by $B_r^{d_{\mathcal{X}}}(x)$ or $B_r(x)$ if the pseudometric $d_{\mathcal{X}}$ is clear from the context. By $\|\cdot\|_p$, $p \in [1, \infty]$, we denote the ℓ^p-norm, and by $\langle \cdot, \cdot \rangle$ the Euclidean inner product of given vectors. By $\|\cdot\|_{\mathrm{op}}$ we denote the operator norm induced by the Euclidean norm and by $\|\cdot\|_F$ the Frobenius norm of given matrices. For $p \in [1, \infty]$, $s \in [0, \infty)$, $d \in \mathbb{N}$, and $\mathcal{X} \subset \mathbb{R}^d$, we denote by $W^{s,p}(\mathcal{X})$ the Sobolev-Slobodeckij space, which for $s = 0$ is just a Lebesgue space, i.e., $W^{0,p}(\mathcal{X}) = L^p(\mathcal{X})$. For measurable spaces $\mathcal{X}$ and $\mathcal{Y}$, we define $\mathcal{M}(\mathcal{X}, \mathcal{Y})$ to be the set of measurable functions from $\mathcal{X}$ to $\mathcal{Y}$. We denote by $\hat{g}$ the Fourier transform[3] of a tempered distribution g. For probabilistic statements, we will assume a suitable underlying probability space with probability measure $\mathbb{P}$. For an $\mathcal{X}$-valued random variable X, we denote by $\mathbb{E}[X]$ and $\mathbb{V}[X]$ its expectation and variance and by $\mathbb{P}_X$ the image measure of X on $\mathcal{X}$, i.e., $\mathbb{P}_X(A) = \mathbb{P}(X \in A)$ for every measurable set $A \subset \mathcal{X}$. If possible, we use the corresponding lowercase letter to denote the realization $x \in \mathcal{X}$ of the random variable X for a given outcome. We write I_d for the d-dimensional identity matrix and, for a set A, we write $\mathbb{1}_A$ for the indicator function of A, i.e., $\mathbb{1}_A(x) = 1$ if $x \in A$ and $\mathbb{1}_A(x) = 0$ else.

1.2 Foundations of learning theory

Before we continue to describe recent developments in the mathematical analysis of deep learning methods, we start by providing a concise overview of the classical mathematical and statistical theory underlying machine learning tasks and algorithms which, in their most general form, can be formulated as follows.

Definition 1.1 (Learning - informal)**.** *Let $\mathcal{X}, \mathcal{Y}$, and $\mathcal{Z}$ be measurable spaces. In a learning task, one is given data in $\mathcal{Z}$ and a loss function $\mathcal{L} \colon \mathcal{M}(\mathcal{X}, \mathcal{Y}) \times \mathcal{Z} \to \mathbb{R}$. The goal is to choose a hypothesis set $\mathcal{F} \subset \mathcal{M}(\mathcal{X}, \mathcal{Y})$ and construct a learning algorithm, i.e., a mapping*

$$\mathcal{A} \colon \bigcup_{m \in \mathbb{N}} \mathcal{Z}^m \to \mathcal{F},$$

that uses training data $s = (z^{(i)})_{i=1}^m \in \mathcal{Z}^m$ to find a model $f_s = \mathcal{A}(s) \in \mathcal{F}$ that performs well on the training data s and also generalizes to unseen data $z \in \mathcal{Z}$. Here, performance is measured via the loss function $\mathcal{L}$ and the corresponding loss $\mathcal{L}(f_s, z)$ and, informally speaking, generalization means that the out-of-sample performance of f_s at z behaves similar to the in-sample performance on s.

Definition 1.1 is deliberately vague on how to measure generalization performance. Later, we will often study the *expected* out-of-sample performance. To talk about expected performance, a data distribution needs to be specified. We will revisit this point in Assumption 1.10 and Definition 1.11.

For simplicity, we focus on one-dimensional, supervised prediction tasks with input features in Euclidean space as defined in the following.

Definition 1.2 (Prediction task)**.** *In a prediction task, we have that $\mathcal{Z} \coloneqq \mathcal{X} \times \mathcal{Y}$, i.e., we are given training data $s = ((x^{(i)}, y^{(i)}))_{i=1}^m$ that consist of input features $x^{(i)} \in \mathcal{X}$ and corresponding labels $y^{(i)} \in \mathcal{Y}$. For one-dimensional regression tasks with $\mathcal{Y} \subset \mathbb{R}$, we consider the quadratic loss $\mathcal{L}(f, (x, y)) = (f(x) - y)^2$ and,*

[3] Respecting common notation, we will also use the hat symbol to denote the minimizer of the empirical risk $\widehat{f}_s$ in Definition 1.8 but this clash of notation does not cause any ambiguity.

for binary classification tasks with $\mathcal{Y} = \{-1, 1\}$, we consider the 0-1 *loss $\mathcal{L}(f, (x, y)) = \mathbb{1}_{(-\infty,0)}(yf(x))$. We assume that our input features are in Euclidean space, i.e., $\mathcal{X} \subset \mathbb{R}^d$ with input dimension $d \in \mathbb{N}$.*

In a prediction task, we aim for a model $f_s \colon \mathcal{X} \to \mathcal{Y}$, such that, for unseen pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, $f_s(x)$ is a good prediction of the true label y. However, note that large parts of the presented theory can be applied to more general settings.

Remark 1.3 (Learning tasks)**.** *Apart from straightforward extensions to multi-dimensional prediction tasks and other loss functions, we want to mention that unsupervised and semi-supervised learning tasks are often treated as prediction tasks. More precisely, one transforms unlabeled training data $z^{(i)}$ into features $x^{(i)} = T_1(z^{(i)}) \in \mathcal{X}$ and labels $y^{(i)} = T_2(z^{(i)}) \in \mathcal{Y}$ using suitable transformations $T_1 \colon \mathcal{Z} \to \mathcal{X}$, $T_2 \colon \mathcal{Z} \to \mathcal{Y}$. In doing so, one asks for a model f_s approximating the transformation $T_2 \circ T_1^{-1} \colon \mathcal{X} \to \mathcal{Y}$ which is, e.g., done in order to learn feature representations or invariances.*

Furthermore, one can consider density estimation tasks, where $\mathcal{X} = \mathcal{Z}$, $\mathcal{Y} \coloneqq [0, \infty]$, and $\mathcal{F}$ consists of probability densities with respect to some σ-finite reference measure μ on $\mathcal{Z}$. One then aims for a probability density f_s that approximates the density of the unseen data z with respect to μ. One can perform $L^2(\mu)$-approximation based on the discretization $\mathcal{L}(f, z) = -2f(z) + \|f\|^2_{L^2(\mu)}$ or maximum likelihood estimation based on the surprisal $\mathcal{L}(f, z) = -\log(f(z))$.

In deep learning the hypothesis set $\mathcal{F}$ consists of *realizations of neural networks* $\Phi_a(\cdot, \theta)$, $\theta \in \mathcal{P}$, with a given *architecture* a and *parameter set* $\mathcal{P}$. In practice, one uses the term neural network for a range of functions that can be represented by directed acyclic graphs, where the vertices correspond to elementary almost everywhere differentiable functions parametrizable by $\theta \in \mathcal{P}$ and the edges symbolize compositions of these functions. In Section 6, we will review some frequently used architectures, in the other sections, however, we will mostly focus on *fully connected feedforward* (FC) neural networks as defined below.

Definition 1.4 (FC neural network)**.** *A fully connected feedforward neural network is given by its architecture $a = (N, \varrho)$, where $L \in \mathbb{N}$, $N \in \mathbb{N}^{L+1}$, and $\varrho \colon \mathbb{R} \to \mathbb{R}$. We refer to ϱ as the activation function, to L as the number of layers, and to N_0, N_L, and N_ℓ, $\ell \in [L-1]$, as the number of neurons in the input, output, and ℓ-th hidden layer, respectively. We denote the number of parameters by*

$$P(N) \coloneqq \sum_{\ell=1}^{L} N_\ell N_{\ell-1} + N_\ell$$

and define the corresponding realization function $\Phi_a \colon \mathbb{R}^{N_0} \times \mathbb{R}^{P(N)} \to \mathbb{R}^{N_L}$ which satisfies for every input $x \in \mathbb{R}^{N_0}$ and parameters

$$\theta = (\theta^{(\ell)})_{\ell=1}^{L} = ((W^{(\ell)}, b^{(\ell)}))_{\ell=1}^{L} \in \bigtimes_{\ell=1}^{L} (\mathbb{R}^{N_\ell \times N_{\ell-1}} \times \mathbb{R}^{N_\ell}) \cong \mathbb{R}^{P(N)}$$

that $\Phi_a(x, \theta) = \Phi^{(L)}(x, \theta)$, where

$$\begin{aligned} \Phi^{(1)}(x, \theta) &= W^{(1)}x + b^{(1)}, \\ \bar{\Phi}^{(\ell)}(x, \theta) &= \varrho\big(\Phi^{(\ell)}(x, \theta)\big), \quad \ell \in [L-1], \quad \text{and} \\ \Phi^{(\ell+1)}(x, \theta) &= W^{(\ell+1)}\bar{\Phi}^{(\ell)}(x, \theta) + b^{(\ell+1)}, \quad \ell \in [L-1], \end{aligned} \tag{1.1}$$

and ϱ is applied componentwise. We refer to $W^{(\ell)} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and $b^{(\ell)} \in \mathbb{R}^{N_\ell}$ as the weight matrices and bias vectors, and to $\bar{\Phi}^{(\ell)}$ and $\Phi^{(\ell)}$ as the activations and pre-activations of the N_ℓ neurons in the ℓ-th layer. The width and depth of the architecture are given by $\|N\|_\infty$ and L and we call the architecture deep if $L > 2$ and shallow if $L = 2$.

The underlying directed acyclic graph of FC networks is given by compositions of the affine linear maps $x \mapsto W^{(\ell)}x + b^{(\ell)}$, $\ell \in [L]$, with the activation function ϱ intertwined, see Figure 1.1. Typical activation

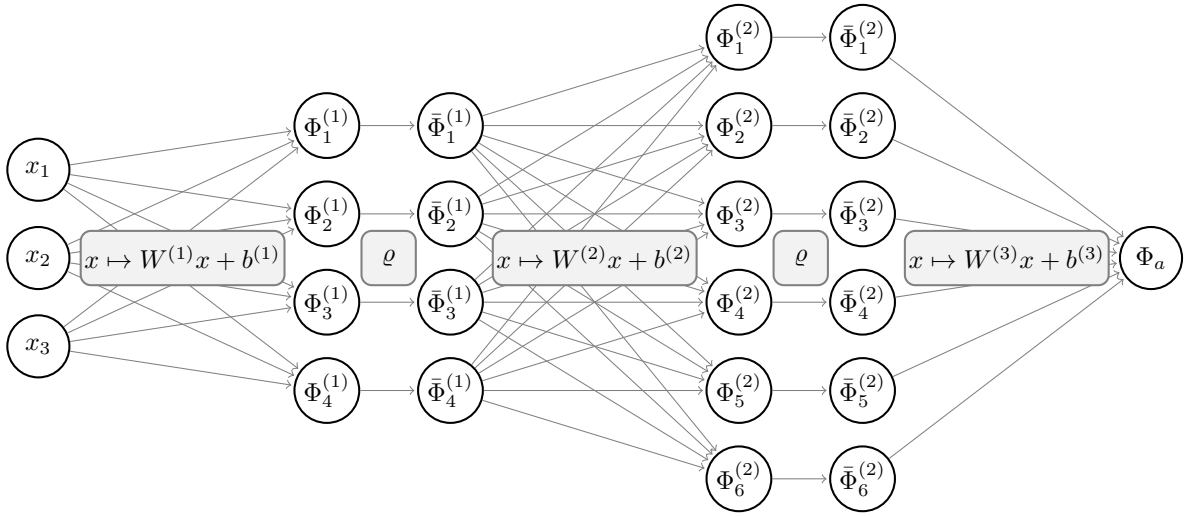


Figure 1.1: Graph (grey) and (pre-)activations of the neurons (white) of a deep fully connected feedforward neural network $\Phi_a \colon \mathbb{R}^3 \times \mathbb{R}^{53} \mapsto \mathbb{R}$ with architecture $a = ((3, 4, 6, 1), \varrho)$ and parameters $\theta = ((W^{(\ell)}, b^{(\ell)})_{\ell=1}^3$.

functions used in practice are variants of the *rectified linear unit* (ReLU) given by $\varrho_R(x) \coloneqq \max\{0, x\}$ and *sigmoidal functions* $\varrho \in C(\mathbb{R})$ satisfying $\varrho(x) \to 1$ for $x \to \infty$ and $\varrho(x) \to 0$ for $x \to -\infty$, such as the logistic function $\varrho_\sigma(x) \coloneqq 1/(1+e^{-x})$ (often referred to as *the* sigmoid function). See also Table 1 for a comprehensive list of widely used activation functions.

Remark 1.5 (Neural networks)**.** *If not further specified, we will use the term (neural) network, or the abbreviation NN, to refer to FC neural networks. Note that many of the architectures used in practice (see Section 6) can be written as special cases of Definition 1.4 where, e.g., specific parameters are prescribed by constants or shared with other parameters. Furthermore, note that affine linear functions are NNs with depth $L = 1$. We will also consider biasless NNs given by linear mappings without bias vector, i.e., $b^{(\ell)} = 0$, $\ell \in [L]$. In particular, any NN can always be written without bias vectors by redefining*

$$x \to \begin{bmatrix} x \\ 1 \end{bmatrix}, \quad (W^{(\ell)}, b^{(\ell)}) \to \begin{bmatrix} W^{(\ell)} & b^{(\ell)} \\ 0 & 1 \end{bmatrix}, \quad \ell \in [L-1], \quad \text{and} \quad (W^{(L)}, b^{(L)}) \to \begin{bmatrix} W^{(L)} & b^{(L)} \end{bmatrix}.$$

To enhance readability we will often not specify the underlying architecture $a = (N, \varrho)$ or the parameters $\theta \in \mathbb{R}^{P(N)}$ and use the term NN to refer to the architecture as well as the realization functions $\Phi_a(\cdot, \theta) \colon \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ or $\Phi_a \colon \mathbb{R}^{N_0} \times \mathbb{R}^{P(N)} \to \mathbb{R}^{N_L}$. However, we want to emphasize that one cannot infer the underlying architecture or properties like magnitude of parameters solely from these functions as the mapping $(a, \theta) \mapsto \Phi_a(\cdot, \theta)$ is highly non-injective. As an example, we can set $W^{(L)} = 0$ which implies $\Phi_a(\cdot, \theta) = b^{(L)}$ for all architectures $a = (N, \varrho)$ and all values of $(W^{(\ell)}, b^{(\ell)})_{\ell=1}^{L-1}$.

In view of our considered prediction tasks in Definition 1.2, this naturally leads to the following hypothesis sets of neural networks.

Definition 1.6 (Hypothesis sets of neural networks)**.** *Let $a = (N, \varrho)$ be a NN architecture with input dimension $N_0 = d$, output dimension $N_L = 1$, and measurable activation function ϱ. For regression tasks the corresponding hypothesis set is given by*

$$\mathcal{F}_a = \left\{ \Phi_a(\cdot, \theta) \colon \theta \in \mathbb{R}^{P(N)} \right\}$$

and for classification tasks by

$$\mathcal{F}_{a,\mathrm{sgn}} = \left\{ \mathrm{sgn}(\Phi_a(\cdot, \theta)) \colon \theta \in \mathbb{R}^{P(N)} \right\}, \quad \text{where} \quad \mathrm{sgn}(x) \coloneqq \begin{cases} 1, & \text{if } x \geq 0, \\ -1, & \text{if } x < 0. \end{cases}$$

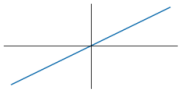
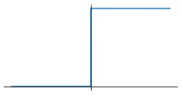
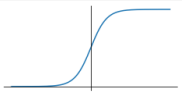
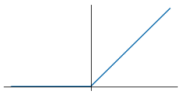
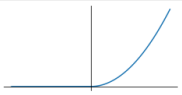
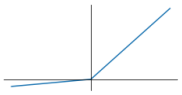
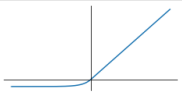
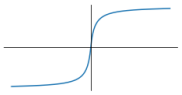
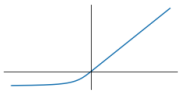
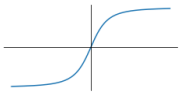
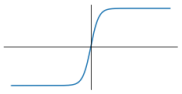
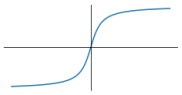
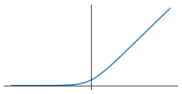
Name	**Given as a function of $x \in \mathbb{R}$ by**	**Plot**
linear	x	
Heaviside / step function	$\mathbb{1}_{(0,\infty)}(x)$	
logistic / sigmoid	$\frac{1}{1+e^{-x}}$	
rectified linear unit (ReLU)	$\max\{0, x\}$	
power rectified linear unit	$\max\{0, x\}^k$ for $k \in \mathbb{N}$	
parametric ReLU (PReLU)	$\max\{ax, x\}$ for $a \geq 0$, $a \neq 1$	
exponential linear unit (ELU)	$x \cdot \mathbb{1}_{[0,\infty)}(x) + (e^x - 1) \cdot \mathbb{1}_{(-\infty,0)}(x)$	
softsign	$\frac{x}{1+\lvert x\rvert}$	
inverse square root linear unit	$x \cdot \mathbb{1}_{[0,\infty)}(x) + \frac{x}{\sqrt{1+ax^2}} \cdot \mathbb{1}_{(-\infty,0)}(x)$ for $a > 0$	
inverse square root unit	$\frac{x}{\sqrt{1+ax^2}}$ for $a > 0$	
tanh	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	
arctan	$\arctan(x)$	
softplus	$\ln(1 + e^x)$	
Gaussian	$e^{-x^2/2}$	

Table 1: List of commonly used activation functions.

Note that we compose the output of the NN with the sign function in order to obtain functions mapping to $\mathcal{Y} = \{-1, 1\}$. This can be generalized to multi-dimensional classification tasks by replacing the sign by an argmax function. Given a hypothesis set, a popular learning algorithm is *empirical risk minimization* (ERM), which minimizes the average loss on the given training data, as described in the next definitions.

Definition 1.7 (Empirical risk)**.** *For training data $s = (z^{(i)})_{i=1}^m \in \mathcal{Z}^m$ and a function $f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$, we define the empirical risk by*

$$\widehat{\mathcal{R}}_s(f) \coloneqq \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(f, z^{(i)}).$$

Definition 1.8 (ERM learning algorithm)**.** *Given a hypothesis set $\mathcal{F}$, an empirical risk minimization algorithm $\mathcal{A}^{\mathrm{erm}}$ chooses[4] for training data $s \in \mathcal{Z}^m$ a minimizer $\widehat{f}_s \in \mathcal{F}$ of the empirical risk in $\mathcal{F}$, i.e.,*

$$\mathcal{A}^{\mathrm{erm}}(s) \in \arg\min_{f \in \mathcal{F}} \widehat{\mathcal{R}}_s(f). \tag{1.2}$$

Remark 1.9 (Surrogate loss and regularization)**.** *Note that, for classification tasks, one needs to optimize over non-differentiable functions with discrete outputs in* (1.2)*. For NN hypothesis sets $\mathcal{F}_{a,\mathrm{sgn}}$ one typically uses the corresponding hypothesis set for regression tasks $\mathcal{F}_a$ to find an approximate minimizer $\widehat{f}_s^{\mathrm{surr}} \in \mathcal{F}_a$ of*

$$\frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^{\mathrm{surr}}(f, z^{(i)}),$$

where $\mathcal{L}^{\mathrm{surr}} \colon \mathcal{M}(\mathcal{X}, \mathbb{R}) \times \mathcal{Z} \to \mathbb{R}$ is a surrogate loss guaranteeing that $\mathrm{sgn}(\widehat{f}_s^{\mathrm{surr}}) \in \arg\min_{f \in \mathcal{F}_{a,\mathrm{sgn}}} \widehat{\mathcal{R}}_s(f)$. A frequently used surrogate loss is the logistic loss[5] given by

$$\mathcal{L}^{\mathrm{surr}}(f, z) = \log\left(1 + e^{-yf(x)}\right).$$

In various learning tasks one also adds regularization terms to the minimization problem in (1.2)*, such as penalties on the norm of the parameters of the NN, i.e.,*

$$\min_{\theta \in \mathbb{R}^{P(N)}} \widehat{\mathcal{R}}_s(\Phi_a(\cdot, \theta)) + \alpha \|\theta\|_2^2,$$

where $\alpha \in (0, \infty)$ is a regularization parameter. Note that in this case the minimizer depends on the chosen parameters θ and not only on the realization function $\Phi_a(\cdot, \theta)$, see also Remark 1.5.

Coming back to our initial, informal description of learning in Definition 1.1, we have now outlined potential learning tasks in Definition 1.2, NN hypothesis sets in Definition 1.6, a metric for the in-sample performance in Definition 1.7, and a corresponding learning algorithm in Definition 1.8. However, we are still lacking a mathematical concept to describe the out-of-sample (generalization) performance of our learning algorithm. This question has been intensively studied in the field of statistical learning theory, see Section 1 for various references.

In this field one usually establishes a connection between unseen data z and the training data $s = (z^{(i)})_{i=1}^m$ by imposing that z and $z^{(i)}$, $i \in [m]$, are realizations of independent samples drawn from the same distribution.

Assumption 1.10 (Independent and identically distributed data)**.** *We assume that $z^{(1)}, \dots, z^{(m)}, z$ are realizations of i.i.d. random variables $Z^{(1)}, \dots, Z^{(m)}, Z$.*

[4] For simplicity, we assume that the minimum is attained which, for instance, is the case if $\mathcal{F}$ is a compact topological space on which $\widehat{\mathcal{R}}_s$ is continuous. Hypothesis sets of NNs $\mathcal{F}_{(N,\varrho)}$ constitute a compact space if, e.g., one chooses a compact parameter set $\mathcal{P} \subset \mathbb{R}^{P(N)}$ and a continuous activation function ϱ. One could also work with approximate minimizers, see [AB99].

[5] This can be viewed as cross-entropy between the label y and the output of f composed with a logistic function ϱ_σ. In a multi-dimensional setting one can replace the logistic function with a softmax function.

In this formal setting, we can compute the average out-of-sample performance of a model. Recall from our notation in Section 1.1 that we denote by $\mathbb{P}_Z$ the image measure of Z on $\mathcal{Z}$, which is the underlying distribution of our training data $S = (Z^{(i)})_{i=1}^m \sim \mathbb{P}_Z^m$ and unknown data $Z \sim \mathbb{P}_Z$.

Definition 1.11 (Risk)**.** *For a function $f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$, we define*[6] *the risk by*

$$\mathcal{R}(f) \coloneqq \mathbb{E}\big[\mathcal{L}(f, Z)\big] = \int_{\mathcal{Z}} \mathcal{L}(f, z) \, \mathrm{d}\mathbb{P}_Z(z).$$

Defining $S \coloneqq (Z^{(i)})_{i=1}^m$, the risk of a model $f_S = \mathcal{A}(S)$ is thus given by $\mathcal{R}(f_S) = \mathbb{E}\big[\mathcal{L}(f_S, Z)|S\big]$.

For prediction tasks, we can write $Z = (X, Y)$, such that the input features and labels are given by an $\mathcal{X}$-valued random variable X and a $\mathcal{Y}$-valued random variable Y, respectively. Note that for classification tasks the risk equals the probability of misclassification

$$\mathcal{R}(f) = \mathbb{E}[\mathbb{1}_{(-\infty,0)}(Yf(X))] = \mathbb{P}[f(X) \neq Y].$$

For noisy data, there might be a positive, lower bound on the risk, i.e., an irreducible error. If the lower bound on the risk is attained, one can also define the notion of an optimal solution to a learning task.

Definition 1.12 (Bayes-optimal function)**.** *A function $f^* \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$ achieving the smallest risk, the so-called Bayes risk*

$$\mathcal{R}^* \coloneqq \inf_{f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})} \mathcal{R}(f),$$

is called a Bayes-optimal function.

For the prediction tasks in Definition 1.2, we can represent the risk of a function with respect to the Bayes risk and compute the Bayes-optimal function, see, e.g., [CZ07, Propositions 1.8 and 9.3].

Lemma 1.1 (Regression and classification risk)**.** *For a regression task with $\mathbb{V}[Y] < \infty$, the risk can be decomposed into*

$$\mathcal{R}(f) = \mathbb{E}\big[(f(X) - \mathbb{E}[Y|X])^2\big] + \mathcal{R}^*, \quad f \in \mathcal{M}(\mathcal{X}, \mathcal{Y}), \tag{1.3}$$

which is minimized by the regression function $f^*(x) = \mathbb{E}[Y|X = x]$. For a classification task, the risk can be decomposed into

$$\mathcal{R}(f) = \mathbb{E}\big[|\mathbb{E}[Y|X]|\mathbb{1}_{(-\infty,0)}(\mathbb{E}[Y|X]f(X))\big] + \mathcal{R}^*, \quad f \in \mathcal{M}(\mathcal{X}, \mathcal{Y}),$$

which is minimized by the Bayes classifier $f^*(x) = \operatorname{sgn}(\mathbb{E}[Y|X = x])$.

As our model f_S is depending on the random training data S, the risk $\mathcal{R}(f_S)$ is a random variable and we might aim[7] for $\mathcal{R}(f_S)$ small with high probability or in expectation over the training data. The challenge for the learning algorithm $\mathcal{A}$ is to minimize the risk by only using training data but without knowing the underlying distribution. One can even show that for every learning algorithm there exists a distribution where convergence of the expected risk of f_S to the Bayes risk is arbitrarily slow with respect to the number of samples m [DGL96, Theorem 7.2].

Theorem 1.13 (No free lunch)**.** *Let $a_m \in (0, \infty)$, $m \in \mathbb{N}$, be a monotonically decreasing sequence with $a_1 \leq 1/16$. Then for every learning algorithm $\mathcal{A}$ of a classification task there exists a distribution $\mathbb{P}_Z$ such that for every $m \in \mathbb{N}$ and training data $S \sim \mathbb{P}_Z^m$ it holds that*

$$\mathbb{E}\big[\mathcal{R}(\mathcal{A}(S))\big] \geq \mathcal{R}^* + a_m.$$

[6] Note that this requires $z \mapsto \mathcal{L}(f, z)$ to be measurable for every $f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$, which is the case for our considered prediction tasks.

[7] In order to make probabilistic statements on $\mathcal{R}(f_S)$ we assume that $\mathcal{R}(f_S)$ is a random variable, i.e., measurable. This is, e.g., the case if $\mathcal{F}$ constitutes a measurable space and $s \mapsto \mathcal{A}(s)$ and $f \to \mathcal{R}|_{\mathcal{F}}$ are measurable.

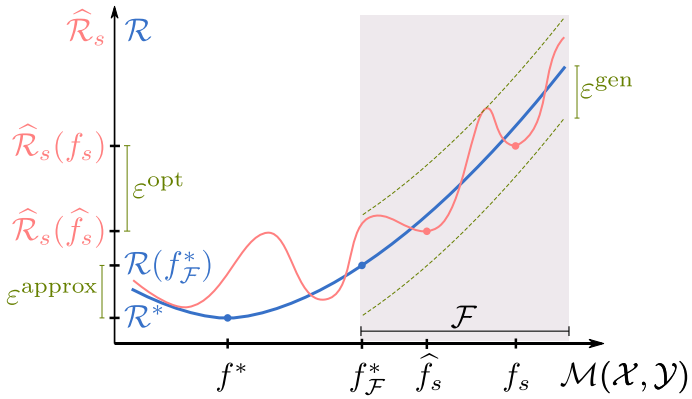


Figure 1.2: Illustration of the errors (A)–(C) in the decomposition of (1.4). It shows an exemplary risk $\widehat{\mathcal{R}}$ (blue) and empirical risk $\widehat{\mathcal{R}}_s$ (red) with respect to the projected space of measurable functions $\mathcal{M}(\mathcal{X},\mathcal{Y})$. Note that the empirical risk and thus ε^{gen} and ε^{opt} depend on the realization $s = (z^{(i)})_{i=1}^m$ of the training data $S \sim \mathbb{P}_Z^m$.

Theorem 1.13 shows the non-existence of a universal learning algorithm for every data distribution $\mathbb{P}_Z$ and shows that useful bounds must necessarily be accompanied by a priori regularity conditions on the underlying distribution $\mathbb{P}_Z$. Such prior knowledge can then be incorporated in the choice of the hypothesis set $\mathcal{F}$. To illustrate this, let $f^*_{\mathcal{F}} \in \arg\min_{f\in\mathcal{F}} \mathcal{R}(f)$ be a best approximation in $\mathcal{F}$, such that we can bound the error

$$\begin{aligned}\mathcal{R}(f_S) - \mathcal{R}^* &= \mathcal{R}(f_S) - \widehat{\mathcal{R}}_S(f_S) + \widehat{\mathcal{R}}_S(f_S) - \widehat{\mathcal{R}}_S(f^*_{\mathcal{F}}) + \widehat{\mathcal{R}}_S(f^*_{\mathcal{F}}) - \mathcal{R}(f^*_{\mathcal{F}}) + \mathcal{R}(f^*_{\mathcal{F}}) - \mathcal{R}^* \\ &\leq \varepsilon^{\text{opt}} + 2\varepsilon^{\text{gen}} + \varepsilon^{\text{approx}}\end{aligned} \tag{1.4}$$

by

(A) an *optimization error* $\varepsilon^{\text{opt}} \coloneqq \widehat{\mathcal{R}}_S(f_S) - \widehat{\mathcal{R}}_S(\widehat{f}_S) \geq \widehat{\mathcal{R}}_S(f_S) - \widehat{\mathcal{R}}_S(f^*_{\mathcal{F}})$, with $\widehat{f}_S$ as in Definition 1.8,

(B) a (uniform[8]) *generalization error* $\varepsilon^{\text{gen}} \coloneqq \sup_{f\in\mathcal{F}} |\mathcal{R}(f) - \widehat{\mathcal{R}}_S(f)| \geq \max\{\mathcal{R}(f_S) - \widehat{\mathcal{R}}_S(f_S), \widehat{\mathcal{R}}_S(f^*_{\mathcal{F}}) - \mathcal{R}(f^*_{\mathcal{F}})\}$, and

(C) an *approximation error* $\varepsilon^{\text{approx}} \coloneqq \mathcal{R}(f^*_{\mathcal{F}}) - \mathcal{R}^*$,

see also Figure 1.2. The approximation error is decreasing when enlarging the hypothesis set, but taking $\mathcal{F} = \mathcal{M}(\mathcal{X},\mathcal{Y})$ prevents controlling the generalization error, see also Theorem 1.13. This suggests a sweet-spot for the complexity of our hypothesis set $\mathcal{F}$ and is usually referred to as the *bias-variance trade-off*, see also Figure 1.4 below. In the next sections, we will sketch mathematical ideas to tackle each of the errors in (A)–(C) in the context of deep learning. Observe that we bound the generalization and optimization error with respect to the empirical risk $\widehat{\mathcal{R}}_S$ and its minimizer $\widehat{f}_S$ which is motivated by the fact that in deep-learning-based applications one typically tries to minimize variants of $\widehat{\mathcal{R}}_S$.

1.2.1 Optimization

The first error in the decomposition of (1.4) is the optimization error: ε^{opt}. This error is primarily influenced by the numerical algorithm $\mathcal{A}$ that is used to find the model f_s in a hypothesis set of NNs for given training data $s \in \mathcal{Z}^m$. We will focus on the typical setting where such an algorithm tries to approximately minimize the empirical risk $\widehat{\mathcal{R}}_s$. While there are many conceivable methods to solve this minimization problem, by far the most common are gradient-based methods. The main reason for the popularity of gradient-based

[8] Although this uniform deviation can be a coarse estimate it is frequently considered to allow for the application of uniform laws of large numbers from the theory of empirical processes.

methods is that for FC networks as in Definition 1.4, the accurate and efficient computation of pointwise derivatives $\nabla_\theta \Phi_a(x,\theta)$ is possible by means of automatic differentiation, a specific form of which is often referred to as the *backpropagation algorithm* [Kel60, Dre62, Lin70, RHW86, GW08]. This numerical scheme is also applicable in general settings, such as, when the architecture of the NN is given by a general directed acyclic graph. Using these pointwise derivatives, one usually attempts to minimize the empirical risk $\widehat{\mathcal{R}}_s$ by updating the parameters θ according to a variant of *stochastic gradient descent* (SGD), which we shall review below in a general formulation:

Algorithm 1: Stochastic gradient descent

Input : Differentiable function $r\colon \mathbb{R}^p \to \mathbb{R}$, sequence of step-sizes $\eta_k \in (0,\infty)$, $k \in [K]$, $\mathbb{R}^p$-valued random variable $\Theta^{(0)}$.

Output : Sequence of $\mathbb{R}^p$-valued random variables $(\Theta^{(k)})_{k=1}^K$.

for $k = 1, \dots, K$ **do**

  Let $D^{(k)}$ be a random variable such that $\mathbb{E}[D^{(k)}|\Theta^{(k-1)}] = \nabla r(\Theta^{(k-1)})$;

  Set $\Theta^{(k)} \coloneqq \Theta^{(k-1)} - \eta_k D^{(k)}$;

end

If $D^{(k)}$ is chosen deterministically in Algorithm 1, i.e., $D^{(k)} = \nabla r(\Theta^{(k-1)})$, then the algorithm is known as *gradient descent*. To minimize the empirical loss, we apply SGD with $r\colon \mathbb{R}^{P(N)} \to \mathbb{R}$ set to $r(\theta) = \widehat{\mathcal{R}}_s(\Phi_a(\cdot,\theta))$. More concretely, one might choose a *batch-size* $m' \in \mathbb{N}$ with $m' \leq m$ and consider the iteration

$$\Theta^{(k)} \coloneqq \Theta^{(k-1)} - \frac{\eta_k}{m'} \sum_{z \in S'} \nabla_\theta \mathcal{L}(\Phi_a(\cdot, \Theta^{(k-1)}), z), \tag{1.5}$$

where S' is a so-called *mini-batch* of size $|S'| = m'$ chosen uniformly[9] at random from the training data s. The sequence of step-sizes $(\eta_k)_{k\in\mathbb{N}}$ is often called *learning rate* in this context. Stopping at step K, the output of a deep learning algorithm $\mathcal{A}$ is then given by

$$f_s = \mathcal{A}(s) = \Phi_a(\cdot, \bar{\theta}),$$

where $\bar{\theta}$ can be chosen to be the realization of the last parameter $\Theta^{(K)}$ of (1.5) or a convex combination of $(\Theta^{(k)})_{k=1}^K$ such as the mean.

Algorithm 1 was originally introduced in [RM51] in the context of finding the root of a nondecreasing function from noisy measurements. Shortly afterwards this idea was applied to find a unique minimum of a Lipschitz-regular function that has no flat regions away from the global minimum [KW52].

In some regimes, we can guarantee convergence of SGD at least in expectation, see [NY83, NJLS09, SSSSS09], [SDR14, Section 5.9], [SSBD14, Chapter 14]. One prototypical convergence guarantee that is found in the aforementioned references in various forms is stated below.

Theorem 1.14 (Convergence of SGD)**.** *Let $p, K \in \mathbb{N}$ and let $r\colon \mathbb{R}^p \supset B_1(0) \to \mathbb{R}$ be differentiable and convex. Further let $(\Theta^{(k)})_{k=1}^K$ be the output of Algorithm 1 with initialization $\Theta^{(0)} = 0$, step-sizes $\eta_k = K^{-1/2}$, $k \in [K]$, and random variables $(D^{(k)})_{k=1}^K$ satisfying that $\|D^{(k)}\|_2 \leq 1$ almost surely for all $k \in [K]$. Then*

$$\mathbb{E}[r(\bar{\Theta})] - r(\theta^*) \leq \frac{1}{\sqrt{K}},$$

where $\bar{\Theta} \coloneqq \frac{1}{K}\sum_{k=1}^K \Theta^{(k)}$ and $\theta^* \in \arg\min_{\theta \in B_1(0)} r(\theta)$.

Theorem 1.14 can be strengthened to yield a faster convergence rate if the convexity is replaced by strict convexity. If r is not convex, then convergence to a global minimum can in general not be guaranteed. In fact, in that case, stochastic gradient descent may converge to a local, non-global minimum, see Figure 1.3 for an example.

[9] We remark that in practice one typically picks S' by selecting a subset of training data in a way to cover the full training data after one *epoch* of $\lceil m/m' \rceil$ many steps. This, however, does not necessarily yield an unbiased estimator $D^{(k)}$ of $\nabla_\theta r(\Theta^{(k-1)})$ given $\Theta^{(k-1)}$.

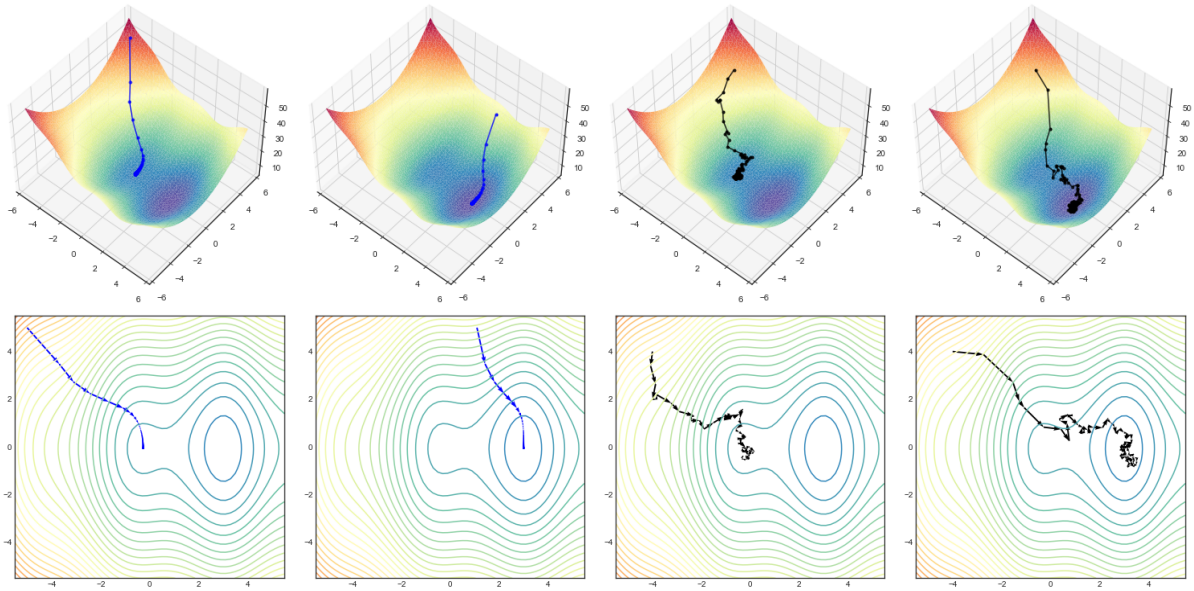


Figure 1.3: Examples of the dynamics of gradient descent (left) and stochastic gradient descent (right) for an objective function with one non-global minimum next to the global minimum. We see that depending on the initial condition and also on fluctuations in the stochastic part of SGD the algorithm can fail or succeed in finding the global minimum.

Moreover, gradient descent, i.e., the deterministic version of Algorithm 1, will stop progressing if at any point the gradient of r vanishes. This is the case in every stationary point of r. A stationary point is either a local minimum, a local maximum, or a saddle point. One would expect that if the direction of the step $D^{(k)}$ in Algorithm 1 is not deterministic, then the random fluctuations may allow the iterates to escape saddle points. Indeed, results guaranteeing convergence to local minima exist under various conditions on the type of saddle points that r admits, [NJLS09, GL13, GHJY15, LSJR16, JKNvW20].

In addition, many methods that improve the convergence by, for example, introducing more elaborate step-size rules or a momentum term have been established. We shall not review these methods here, but instead refer to [GBC16, Chapter 8] for an overview.

1.2.2 Approximation

Generally speaking, NNs, even FC NNs (see Definition 1.4) with only $L = 2$ layers, are universal approximators, meaning that under weak conditions on the activation function ϱ they can approximate any continuous function on a compact set up to arbitrary precision [Cyb89, Fun89, HSW89, LLPS93].

Theorem 1.15 (Universal approximation theorem)**.** *Let $d \in \mathbb{N}$, let $K \subset \mathbb{R}^d$ be compact, and let $\varrho \in L^\infty_{\mathrm{loc}}(\mathbb{R})$ be an activation function such that the closure of the points of discontinuity of ϱ is a Lebesgue null set. Further let*

$$\widetilde{\mathcal{F}} \coloneqq \bigcup_{n \in \mathbb{N}} \mathcal{F}_{((d,n,1),\varrho)}$$

be the corresponding set of two-layer NN realizations. Then it holds that $C(K) \subset \mathrm{cl}(\widetilde{\mathcal{F}})$ (where the closure is taken with respect to the topology induced by the $L^\infty(K)$-norm) if and only if there does not exist a polynomial $p\colon \mathbb{R} \to \mathbb{R}$ with $p = \varrho$ almost everywhere.

The theorem can be proven by the theorem of Hahn–Banach, which implies that $\widetilde{\mathcal{F}}$ being dense in some

real normed vector space $\mathcal{S}$ is equivalent to the following condition: For all non-trivial functionals $F \in \mathcal{S}' \setminus \{0\}$ from the topological dual space of $\mathcal{S}$ there exist parameters $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that

$$F(\varrho(\langle w, \cdot \rangle + b)) \neq 0.$$

In case of $\mathcal{S} = C(K)$ we have by the Riesz–Markov–Kakutani representation theorem that $\mathcal{S}'$ is the space of signed Borel measures on K, see [Rud06]. Therefore, Theorem 1.15 holds, if ϱ is such that, for a signed Borel measure μ,

$$\int_K \varrho(\langle w, x \rangle + b) \, \mathrm{d}\mu(x) = 0 \tag{1.6}$$

for all $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ implies that $\mu = 0$. An activation function ϱ satisfying this condition is called *discriminatory*. It is not hard to see that any sigmoidal ϱ is discriminatory. Indeed, assume that ϱ satisfies (1.6) for all $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$. Since for every $x \in \mathbb{R}^d$ it holds that $\varrho(ax+b) \to \mathbb{1}_{(0,\infty)}(x) + \varrho(b)\mathbb{1}_{\{0\}}(x)$ for $a \to \infty$, we conclude by superposition and passing to the limit that for all $c_1, c_2 \in \mathbb{R}$ and $w \in \mathbb{R}^d$, $b \in \mathbb{R}$

$$\int_K \mathbb{1}_{[c_1,c_2]}(\langle w, x \rangle + b) \, \mathrm{d}\mu(x) = 0.$$

Representing the exponential function $x \mapsto e^{-2\pi i x}$ as the limit of sums of elementary functions yields that $\int_K e^{-2\pi i(\langle w,x\rangle + b)} \, \mathrm{d}\mu(x) = 0$ for all $w \in \mathbb{R}^d$, $b \in \mathbb{R}$. Hence, the Fourier transform of μ vanishes which implies that $\mu = 0$.

Theorem 1.15 addresses a uniform approximation problem on a general compact set. If we are given a finite number of points and only care about good approximation at these points, then one can ask if this approximation problem is potentially simpler. Below we see that, if the number of neurons is larger or equal to the number of data points, then one can always interpolate, i.e., exactly fit the data on a given finite number of points.

Proposition 1.1 (Interpolation)**.** *Let $d, m \in \mathbb{N}$, let $x^{(i)} \in \mathbb{R}^d$, $i \in [m]$, with $x^{(i)} \neq x^{(j)}$ for $i \neq j$, let $\varrho \in C(\mathbb{R})$, and assume that ϱ is not a polynomial. Then, there exist parameters $\theta^{(1)} \in \mathbb{R}^{m \times d} \times \mathbb{R}^m$ with the following property: For every $k \in \mathbb{N}$ and every sequence of labels $y^{(i)} \in \mathbb{R}^k$, $i \in [m]$, there exist parameters $\theta^{(2)} = (W^{(2)}, 0) \in \mathbb{R}^{k \times m} \times \mathbb{R}^k$ for the second layer of the NN architecture $a = ((d, m, k), \varrho)$ such that*

$$\Phi_a(x^{(i)}, (\theta^{(1)}, \theta^{(2)})) = y^{(i)}, \quad i \in [m].$$

Let us sketch the proof in the following. First, note that Theorem 1.15 also holds for functions $g \in C(K, \mathbb{R}^m)$ with multi-dimensional output by approximating each one-dimensional component $x \mapsto (g(x))_i$ and stacking the resulting networks. Second, one can add an additional row containing only zeros to the weight matrix $W^{(1)}$ of the approximating neural network as well as an additional entry to the vector $b^{(1)}$. The effect of this is that we obtain an additional neuron with constant output. Since $\varrho \neq 0$, we can choose $b^{(1)}$ such that the output of this neuron is not zero. Therefore, we can include the bias vector $b^{(2)}$ of the second layer into the weight matrix $W^{(2)}$, see also Remark 1.5. Now choose $g \in C(\mathbb{R}^m, \mathbb{R}^m)$ to be a function satisfying $g(x^{(i)}) = e^{(i)}$, $i \in [m]$, where $e^{(i)} \in \mathbb{R}^m$ denotes the i-th standard basis vector. By the discussion before there exists a neural network architecture $\tilde{a} = ((d, n, m), \varrho)$ and parameters $\tilde{\theta} = ((\widetilde{W}^{(1)}, \tilde{b}^{(1)}), (\widetilde{W}^{(2)}, 0))$ such that

$$\|\Phi_{\tilde{a}}(\cdot, \tilde{\theta}) - g\|_{L^\infty(K)} < \frac{1}{m}, \tag{1.7}$$

where K is a compact set with $x^{(i)} \in K$, $i \in [m]$. Let us abbreviate the output of the activations in the first layer evaluated at the input features by

$$\widetilde{A} \coloneqq \left[\varrho(\widetilde{W}^{(1)}(x^{(1)}) + \tilde{b}^{(1)})) \dots \varrho(\widetilde{W}^{(1)}(x^{(m)}) + \tilde{b}^{(1)}))\right] \in \mathbb{R}^{n \times m}.$$

The equivalence of the max and operator norm and (1.7) establish that

$$\|\widetilde{W}^{(2)}\widetilde{A} - \mathrm{I}_m\|_{\mathrm{op}} \leq m \max_{i,j \in [m]} \left|(\widetilde{W}^{(2)}\widetilde{A} - \mathrm{I}_m)_{i,j}\right| = m \max_{j \in [m]} \|\Phi_{\tilde{a}}(x^{(j)}, \tilde{\theta}) - g(x^{(j)})\|_\infty < 1,$$

where I_m denotes the $m \times m$ identity matrix. Thus, the matrix $\widetilde{W}^{(2)}\widetilde{A} \in \mathbb{R}^{m\times m}$ needs to have full rank and we can extract m linearly independent rows from $\widetilde{A}$ resulting in an invertible matrix $A \in \mathbb{R}^{m\times m}$. Now, we define the desired parameters $\theta^{(1)}$ for the first layer by extracting the corresponding rows from $\widetilde{W}^{(1)}$ and $\tilde{b}^{(1)}$ and the parameters $\theta^{(2)}$ of the second layer by

$$W^{(2)} \coloneqq \big[y^{(1)} \dots y^{(m)}\big]\, A^{-1} \in \mathbb{R}^{k\times m}.$$

This proves that with any discriminatory activation function we can interpolate arbitrary training data $(x^{(i)}, y^{(i)}) \in \mathbb{R}^d \times \mathbb{R}^k$, $i \in [m]$, using a two-layer NN with m hidden neurons, i.e., $\mathcal{O}(m(d+k))$ parameters.

One can also first project the input features to a one-dimensional line where they are separated and then apply Proposition 1.1 with $d = 1$. For nearly all activation functions, this can be represented by a three-layer NN using only $\mathcal{O}(d + mk)$ parameters[10].

Beyond interpolation results, one can obtain a quantitative version of Theorem 1.15 if one knows additional regularity properties of the Bayes optimal function f^*, such as smoothness, compositionality, and symmetries. For surveys on such results, we refer the reader to [DHP20, GRK20]. For instructive purposes, we review one such result, which can be found in [Mha96, Theorem 2.1], below:

Theorem 1.16 (Approximation of smooth functions)**.** *Let $d, k \in \mathbb{N}$ and $p \in [1,\infty]$. Further let $\varrho \in C^\infty(\mathbb{R})$ and assume that ϱ is not a polynomial. Then there exists a constant $c \in (0,\infty)$ with the following property: For every $n \in \mathbb{N}$ there exist parameters $\theta^{(1)} \in \mathbb{R}^{n\times d} \times \mathbb{R}^n$ for the first layer of the NN architecture $a = ((d, n, 1), \varrho)$ such that for every $g \in W^{k,p}((0,1)^d)$ it holds that*

$$\inf_{\theta^{(2)}\in\mathbb{R}^{1\times n}\times\mathbb{R}} \|\Phi_a(\cdot,(\theta^{(1)},\theta^{(2)})) - g\|_{L^p((0,1)^d)} \le cn^{-\frac{d}{k}}\|g\|_{W^{k,p}((0,1)^d)}.$$

Theorem 1.16 shows that NNs achieve the same optimal approximation rates that, for example, spline-based approximation yields for smooth functions. The idea behind this theorem is based on a strategy that is employed repeatedly throughout the literature. This is the idea of re-approximating classical approximation methods by NNs and thereby transferring the approximation rates of these methods to NNs. In the example of Theorem 1.16, approximation by polynomials is used. The idea is that due to the non-vanishing derivatives of the activation function[11], one can approximate every univariate polynomial via divided differences of the activation function. Specifically, accepting unbounded parameter magnitudes, for any activation function $\varrho\colon \mathbb{R} \to \mathbb{R}$ which is *p-times differentiable* at some point $\lambda \in \mathbb{R}$ with $\varrho^{(p)}(\lambda) \neq 0$, one can approximate the monomial $x \mapsto x^p$ on a compact set $K \subset \mathbb{R}$ up to arbitrary precision by a fixed-size NN via rescaled p-th order difference quotients as

$$\lim_{h\to 0}\sup_{x\in K}\Big|\sum_{i=0}^{p}\frac{(-1)^i\binom{p}{i}}{h^p\varrho^{(p)}(\lambda)}\varrho\big((p/2-i)hx+\lambda\big) - x^p\Big| = 0. \tag{1.8}$$

Let us end this subsection by clarifying the connection of the approximation results above to the error decomposition of (1.4). Consider, for simplicity, a regression task with quadratic loss. Then, the approximation error $\varepsilon^{\text{approx}}$ equals a common L^2-error

$$\begin{aligned}\varepsilon^{\text{approx}} = \mathcal{R}(f_{\mathcal{F}}^*) - \mathcal{R}^* &\overset{(*)}{=} \int_{\mathcal{X}} (f_{\mathcal{F}}^*(x) - f^*(x))^2 \,\mathrm{d}\mathbb{P}_X(x)\\ &\overset{(*)}{=} \min_{f\in\mathcal{F}} \|f - f^*\|^2_{L^2(\mathbb{P}_X)}\\ &\le \min_{f\in\mathcal{F}} \|f - f^*\|^2_{L^\infty(\mathcal{X})},\end{aligned}$$

where the identities marked by $(*)$ follow from Lemma 1.1. Hence, Theorem 1.15 postulates that $\varepsilon^{\text{approx}} \to 0$ for increasing NN sizes, whereas Theorem 1.16 additionally explains how fast $\varepsilon^{\text{approx}}$ converges to 0.

[10] To avoid the $m \times d$ weight matrix (without using shared parameters as in [ZBH+17]) one interjects an approximate one-dimensional identity [PV18, Definition 2.5], which can be arbitrarily well approximated by a NN with architecture $a = ((1, 2, 1), \varrho)$ given that $\varrho'(\lambda) \neq 0$ for some $\lambda \in \mathbb{R}$, see (1.8) below.

[11] The Baire category theorem ensures that for a non-polynomial $\varrho \in C^\infty(\mathbb{R})$ there exists $\lambda \in \mathbb{R}$ with $\varrho^{(p)}(\lambda) \neq 0$ for all $p \in \mathbb{N}$, see, e.g., [Don69, Chapter 10].

1.2.3 Generalization

Towards bounding the generalization error $\varepsilon^{\text{gen}} = \sup_{f\in\mathcal{F}} |\mathcal{R}(f) - \widehat{\mathcal{R}}_S(f)|$, one observes that, for every $f \in \mathcal{F}$, Assumption 1.10 ensures that $\mathcal{L}(f, Z^{(i)})$, $i \in [m]$, are i.i.d. random variables. Thus, one can make use of concentration inequalities to bound the deviation of the empirical risk $\widehat{\mathcal{R}}_S(f) = \frac{1}{m}\sum_{i=1}^m \mathcal{L}(f, Z^{(i)})$ from its expectation $\mathcal{R}(f)$. For instance, assuming boundedness[12] of the loss, Hoeffding's inequality [Hoe63] and a union bound directly imply the following generalization guarantee for countable, weighted hypothesis sets $\mathcal{F}$, see, e.g., [BBL03].

Theorem 1.17 (Generalization bound for countable, weighted hypothesis sets)**.** *Let $m \in \mathbb{N}$, $\delta \in (0,1)$ and assume that $\mathcal{F}$ is countable. Further let p be a probability distribution on $\mathcal{F}$ and assume that $\mathcal{L}(f, Z) \in [0,1]$ almost surely for every $f \in \mathcal{F}$. Then with probability $1-\delta$ (with respect to repeated sampling of $\mathbb{P}_Z^m$-distributed training data S) it holds for every $f \in \mathcal{F}$ that*

$$|\mathcal{R}(f) - \widehat{\mathcal{R}}_S(f)| \leq \sqrt{\frac{\ln(1/p(f)) + \ln(2/\delta)}{2m}}.$$

While the weighting p needs to be chosen before seeing the training data, one could incorporate prior information on the learning algorithm $\mathcal{A}$. For finite hypothesis sets without prior information, setting $p(f) = 1/|\mathcal{F}|$ for every $f \in \mathcal{F}$, Theorem 1.17 implies that, with high probability, it holds that

$$\varepsilon^{\text{gen}} \lesssim \sqrt{\frac{\ln(|\mathcal{F}|)}{m}}. \tag{1.9}$$

Again, one notices that, in line with the bias-variance trade-off, the generalization bound is increasing with the size of the hypothesis set $|\mathcal{F}|$. Although in practice the parameters $\theta \in \mathbb{R}^{P(N)}$ of a NN are discretized according to floating-point arithmetic, the corresponding quantities $|\mathcal{F}_a|$ or $|\mathcal{F}_{a,\text{sgn}}|$ would be huge and we need to find a replacement for the finiteness condition.

We will focus on binary classification tasks and present a main result of VC theory which is to a great extent derived from the work of Vladimir Vapnik and Alexey Chervonenkis [VC71]. While in (1.9) we counted the number of functions in $\mathcal{F}$, we now refine this analysis to the number of functions restricted to a finite subset of $\mathcal{X}$, given by the *growth function*

$$\text{growth}(m, \mathcal{F}) \coloneqq \max_{(x^{(i)})_{i=1}^m \in \mathcal{X}^m} |\{f|_{(x^{(i)})_{i=1}^m} : f \in \mathcal{F}\}|.$$

The growth function can be interpreted as the maximal number of classification patterns in $\{-1,1\}^m$ which functions in $\mathcal{F}$ can realize on m points and thus $\text{growth}(m, \mathcal{F}) \leq 2^m$. The asymptotic behavior of the growth function is determined by a single intrinsic dimension of our hypothesis set $\mathcal{F}$, the so-called *VC-dimension*

$$\text{VCdim}(\mathcal{F}) \coloneqq \sup\big\{m \in \mathbb{N} \cup \{0\} \colon \text{growth}(m, \mathcal{F}) = 2^m\big\},$$

which defines the largest number of points such that $\mathcal{F}$ can realize any classification pattern, see, e.g., [AB99, BBL03]. There exist various results on VC-dimensions of NNs with different activation functions, see, for instance, [BH89, KM97, BMM98, Sak99]. We present the result of [BMM98] for piecewise polynomial activation functions ϱ. It establishes a bound on the VC-dimension of hypothesis sets of NNs for classification tasks $\mathcal{F}_{(N,\varrho),\text{sgn}}$ that scales, up to logarithmic factors, linear in the number of parameters $P(N)$ and quadratic in the number of layers L.

Theorem 1.18 (VC-dimension of neural network hypothesis sets)**.** *Let ϱ be a piecewise polynomial activation function. Then there exists a constant $c \in (0,\infty)$ such that for every $L \in \mathbb{N}$ and $N \in \mathbb{N}^{L+1}$ it holds that*

$$\text{VCdim}(\mathcal{F}_{(N,\varrho),\text{sgn}}) \leq c\big(P(N)L\log(P(N)) + P(N)L^2\big).$$

[12] Note that for our classification tasks in Definition 1.2 it holds that $\mathcal{L}(f, Z) \in \{0,1\}$ for every $f \in \mathcal{F}$. For the regression tasks, one typically assumes boundedness conditions, such as $|Y| \leq c$ and $\sup_{f\in\mathcal{F}} |f(X)| \leq c$ almost surely for some $c \in (0,\infty)$, which yields that $\sup_{f\in\mathcal{F}} |\mathcal{L}(f, Z)| \leq 4c^2$.

Given $(x^{(i)})_{i=1}^m \in \mathcal{X}^m$, there exists a partition of $\mathbb{R}^{P(N)}$ such that $\Phi(x^{(i)}, \cdot)$, $i \in [m]$, are polynomials on each region of the partition. The proof of Theorem 1.18 is based on bounding the number of such regions and the number of classification patterns of a set of polynomials.

A finite VC-dimension ensures the following generalization bound [Tal94, AB99]:

Theorem 1.19 (VC-dimension generalization bound)**.** *There exists a constant $c \in (0, \infty)$ with the following property: For every classification task as in Definition 1.2, every $\mathcal{Z}$-valued random variable Z, and every $m \in \mathbb{N}$, $\delta \in (0,1)$ it holds with probability $1-\delta$ (with respect to repeated sampling of $\mathbb{P}_Z^m$-distributed training data S) that*

$$\sup_{f \in \mathcal{F}} |\mathcal{R}(f) - \widehat{\mathcal{R}}_S(f)| \leq c\sqrt{\frac{\mathrm{VCdim}(\mathcal{F}) + \log(1/\delta))}{m}}.$$

In summary, using NN hypothesis sets $\mathcal{F}_{(N,\varrho),\mathrm{sgn}}$ with a fixed depth and piecewise polynomial activation ϱ for a classification task, with high probability it holds that

$$\varepsilon^{\mathrm{gen}} \lesssim \sqrt{\frac{P(N)\log(P(N))}{m}}. \tag{1.10}$$

In the remainder of this section we will sketch a proof of Theorem 1.19 and, in doing so, present further concepts and complexity measures connected to generalization bounds. We start by observing that McDiarmid's inequality [McD89] ensures that $\varepsilon^{\mathrm{gen}}$ is sharply concentrated around its expectation, i.e., with probability $1-\delta$ it holds that[13]

$$\big|\varepsilon^{\mathrm{gen}} - \mathbb{E}\big[\varepsilon^{\mathrm{gen}}\big]\big| \lesssim \sqrt{\frac{\log(1/\delta)}{m}}. \tag{1.11}$$

To estimate the expectation of the uniform generalization error we employ a *symmetrization argument* [GZ84]. Define $\mathcal{G} \coloneqq \mathcal{L} \circ \mathcal{F} \coloneqq \{\mathcal{L}(f, \cdot) \colon f \in \mathcal{F}\}$, let $\widetilde{S} = (\widetilde{Z}^{(i)})_{i=1}^m \sim \mathbb{P}_Z^m$ be a test data set independent of S, and note that $\mathcal{R}(f) = \mathbb{E}[\widehat{\mathcal{R}}_{\widetilde{S}}(f)]$. By properties of the conditional expectation and Jensen's inequality it holds that

$$\begin{aligned}
\mathbb{E}\big[\varepsilon^{\mathrm{gen}}\big] = \mathbb{E}\Big[\sup_{f\in\mathcal{F}} |\mathcal{R}(f) - \widehat{\mathcal{R}}_S(f)|\Big] &= \mathbb{E}\Big[\sup_{g\in\mathcal{G}} \frac{1}{m}\Big|\sum_{i=1}^m \mathbb{E}\big[g(\widetilde{Z}^{(i)}) - g(Z^{(i)})|S\big]\Big|\Big] \\
&\leq \mathbb{E}\Big[\sup_{g\in\mathcal{G}} \frac{1}{m}\Big|\sum_{i=1}^m g(\widetilde{Z}^{(i)}) - g(Z^{(i)})\Big|\Big] \\
&= \mathbb{E}\Big[\sup_{g\in\mathcal{G}} \frac{1}{m}\Big|\sum_{i=1}^m \tau_i\big(g(\widetilde{Z}^{(i)}) - g(Z^{(i)})\big)\Big|\Big] \\
&\leq 2\mathbb{E}\Big[\sup_{g\in\mathcal{G}} \frac{1}{m}\Big|\sum_{i=1}^m \tau_i g(Z^{(i)})\Big|\Big],
\end{aligned}$$

where we used that multiplications with Rademacher variables $(\tau_1, \dots, \tau_m) \sim \mathcal{U}(\{-1,1\}^m)$ only amount to interchanging $Z^{(i)}$ with $\widetilde{Z}^{(i)}$ which has no effect on the expectation, since $Z^{(i)}$ and $\widetilde{Z}^{(i)}$ have the same distribution. The quantity

$$\mathfrak{R}_m(\mathcal{G}) \coloneqq \mathbb{E}\Big[\sup_{g\in\mathcal{G}} \Big|\frac{1}{m}\sum_{i=1}^m \tau_i g(Z^{(i)})\Big|\Big]$$

is called the *Rademacher complexity*[14] of $\mathcal{G}$. One can also prove a corresponding lower bound [vdVW97], i.e.,

$$\mathfrak{R}_m(\mathcal{G}) - \frac{1}{\sqrt{m}} \lesssim \mathbb{E}\big[\varepsilon^{\mathrm{gen}}\big] \lesssim \mathfrak{R}_m(\mathcal{G}). \tag{1.12}$$

[13] For precise conditions to ensure that the expectation of $\varepsilon^{\mathrm{gen}}$ is well-defined, we refer the reader to [vdVW97, Dud14].

[14] Due to our decomposition in (1.4), we want to uniformly bound the absolute value of the difference between the risk and the empirical risk. It is also common to just bound $\sup_{f\in\mathcal{F}} \mathcal{R}(f) - \widehat{\mathcal{R}}_S(f)$ leading to a definition of the Rademacher complexity without the absolute values which can be easier to deal with.

Now we use a *chaining method* to bound the Rademacher complexity of $\mathcal{F}$ by covering numbers on different scales. Specifically, Dudley's entropy integral [Dud67, LT91] implies that

$$\mathfrak{R}_m(\mathcal{G}) \lesssim \mathbb{E}\Big[\int_0^\infty \sqrt{\frac{\log N_\alpha(\mathcal{G}, d_S)}{m}}\,\mathrm{d}\alpha\Big], \tag{1.13}$$

where

$$N_\alpha(\mathcal{G}, d_S) \coloneqq \inf\Big\{|G| \colon G \subset \mathcal{G},\, \mathcal{G} \subset \bigcup_{g\in G} B_\alpha^{d_S}(g)\Big\}$$

denotes the covering number with respect to the (random) pseudometric given by

$$d_S(f,g) = d_{(Z^{(i)})_{i=1}^m}(f,g) \coloneqq \sqrt{\frac{1}{m}\sum_{i=1}^m \big(f(Z^{(i)}) - g(Z^{(i)})\big)^2}.$$

For the 0-1 loss $\mathcal{L}(f,z) = \mathbb{1}_{(-\infty,0)}(yf(x)) = (1-f(x)y)/2$, we can get rid of the loss function by the fact that

$$N_\alpha(\mathcal{G}, d_S) = N_{2\alpha}(\mathcal{F}, d_{(X^{(i)})_{i=1}^m}). \tag{1.14}$$

The proof is completed by combining the inequalities in (1.11), (1.12), (1.13) and (1.14) with a result of David Haussler [Hau95] which shows that for $\alpha \in (0,1)$ we have

$$\log(N_\alpha(\mathcal{F}, d_{(X^{(i)})_{i=1}^m})) \lesssim \mathrm{VCdim}(\mathcal{F}) \log(1/\alpha). \tag{1.15}$$

We remark that this resembles a typical behavior of covering numbers. For instance, the logarithm of the covering number $\log(N_\alpha(\mathcal{M}))$ of a compact d-dimensional Riemannian manifold $\mathcal{M}$ essentially scales like $d\log(1/\alpha)$. Finally, note that there exists a similar bound to the one in (1.15) for bounded regression tasks making use of the so-called *fat-shattering dimension* [MV03, Theorem 1].

1.3 Do we need a new theory?

Despite the already substantial insight that the classical theories provide, a lot of open questions remain. We will outline these questions below. The remainder of this article then collects modern approaches to explain the following issues:

Why do large neural networks not overfit? In Subsection 1.2.2, we have observed that three-layer NNs with commonly used activation functions and only $\mathcal{O}(d+m)$ parameters can interpolate any training data $(x^{(i)}, y^{(i)}) \in \mathbb{R}^d \times \mathbb{R}$, $i \in [m]$. While this specific representation might not be found in practice, [ZBH$^+$17] indeed trained convolutional[15] NNs with ReLU activation function and about 1.6 million parameters to achieve zero empirical risk on $m = 50000$ training images of the CIFAR10 dataset [KH09] with 32×32 pixels per image, i.e., $d = 1024$. For such large NNs, generalization bounds scaling with the number of parameters $P(N)$ as the VC-dimension bound in (1.10) are vacuous. However, they observed close to state-of-the-art generalization performance[16].

Generally speaking, NNs in practice are observed to generalize well despite having more parameters than training samples (usually referred to as *overparametrization*) and approximately interpolating the training data (usually referred to as *overfitting*). As we cannot perform any better on the training data, there is no trade-off between fit to training data and complexity of the hypothesis set $\mathcal{F}$ happening, seemingly contradicting the classical bias-variance trade-off of statistical learning theory. This is quite surprising, especially given the following additional empirical observations in this regime, see [NTS14, ZBH$^+$17, NBMS17, BHMM19, NKB$^+$20]:

[15] The basic definition of a convolutional NN will be given in Section 6. In [ZBH$^+$17] more elaborate versions such as an *Inception* architecture [SLJ$^+$15] are employed.

[16] In practice one usually cannot measure the risk $\mathcal{R}(f_s)$ and instead evaluates the performance of a trained model f_s by $\widehat{\mathcal{R}}_{\tilde{s}}(f_s)$ using test data $\tilde{s}$, i.e., realizations of i.i.d. random variables distributed according to $\mathbb{P}_Z$ and drawn independently of the training data. In this context one often calls $\mathcal{R}_s(f_s)$ the *training error* and $\mathcal{R}_{\tilde{s}}(f_s)$ the *test error*.

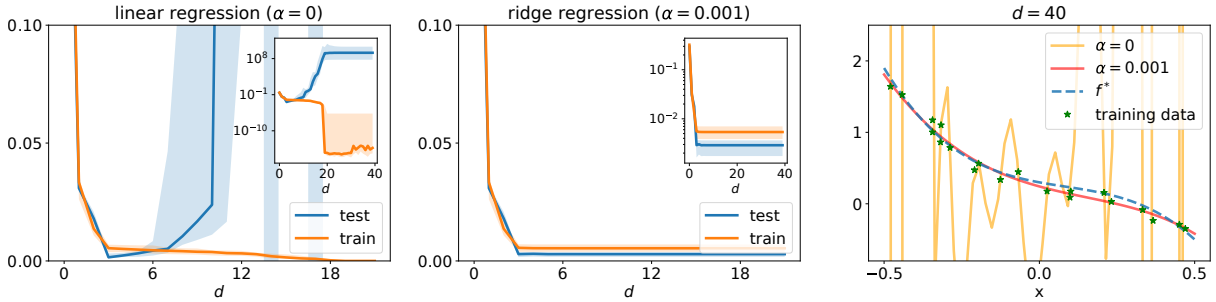


Figure 1.4: The first plot (and its semi-log inset) shows median and interquartile range of the test and training errors of ten independent linear regressions with $m = 20$ samples, polynomial input features $X = (1, Z, \dots, Z^d)$ of degree $d \in [40]$, and labels $Y = f^*(Z) + \nu$, where $Z \sim \mathcal{U}([-0.5, 0.5])$, f^* is a polynomial of degree three, and $\nu \sim \mathcal{N}(0, 0.01)$. This clearly reflects the classical u-shaped bias-variance curve with a sweet-spot at $d = 3$ and drastic overfitting beyond the interpolation threshold at $d = 20$. However, the second plot shows that we can control the complexity of our hypothesis set of linear models by restricting the Euclidean norm of their parameters using ridge regression with a small regularization parameter $\alpha = 10^{-3}$, i.e., minimizing the regularized empirical risk $\frac{1}{m}\sum_{i=1}^{m}(\Phi(X^{(i)}, \theta) - Y^{(i)})^2 + \alpha\|\theta\|_2^2$, where $\Phi(\cdot, \theta) = \langle \theta, \cdot \rangle$. Corresponding examples of $\widehat{f}_s$ are depicted in the last plot.

1. *Zero training error on random labels:* Zero empirical risk can also be achieved for random labels using the same architecture and training scheme with only slightly increased training time: This suggests that the considered hypothesis set of NNs $\mathcal{F}$ can fit arbitrary binary labels, which would imply that $\mathrm{VCdim}(\mathcal{F}) \approx m$ or $\mathfrak{R}_m(\mathcal{F}) \approx 1$ rendering our uniform generalization bounds in Theorem 1.19 and in (1.12) vacuous.

2. *Lack of explicit regularization:* The test error depends only mildly on explicit regularization like norm-based penalty terms or dropout (see [Gér17] for an explanation of different regularization methods): As such regularization methods are typically used to decrease the complexity of $\mathcal{F}$, one might ask if there is any *implicit* regularization (see Figure 1.4), constraining the range of our learning algorithm $\mathcal{A}$ to some smaller, potentially data-dependent subset, i.e., $\mathcal{A}(s) \in \widetilde{\mathcal{F}}_s \subsetneq \mathcal{F}$.

3. *Dependence on the optimization:* The same NN trained to zero empirical risk using different variants of SGD or starting from different initializations can exhibit different test errors: This indicates that the dynamics of gradient descent and properties of the local neighborhood around the model $f_s = \mathcal{A}(s)$ might be correlated with generalization performance.

4. *Interpolation of noisy training data:* One still observes low test error when training up to approximately zero empirical risk using a regression (or surrogate) loss on noisy training data. This is particularly interesting, as the noise is captured by the model but seems not to hurt generalization performance.

5. *Further overparametrization improves generalization performance:* Further increasing the NN size can lead to even lower test error: Together with the previous item, this might ask for a different treatment of models complex enough to fit the training data. According to the traditional lore "The training error tends to decrease whenever we increase the model complexity, that is, whenever we fit the data harder. However with too much fitting, the model adapts itself too closely to the training data, and will not generalize well (i.e., have large test error)", [HTF01]. While this flawlessly describes the situation for certain machine learning tasks (see Figure 1.4), it seems not to be directly applicable here.

In summary, this suggests that the generalization performance of NNs depends on an interplay of the data distribution $\mathbb{P}_Z$ combined with properties of the learning algorithm $\mathcal{A}$, such as the optimization procedure and its range. In particular, classical uniform bounds as in Item (B) of our error decomposition might only

deliver insufficient explanation, see also [NK19]. The mismatch between predictions of classical theory and the practical generalization performance of deep NNs is often referred to as *generalization puzzle*. In Section 2 we will present possible explanations for this phenomenon.

What is the role of depth? We have seen in Subsection 1.2.2 that NNs can closely approximate every function if they are sufficiently wide [Cyb89, Fun89, HSW89]. There are additional classical results that even provide a trade-off between the width and the approximation accuracy [CLM94, Mha96, MP99]. In these results, the central concept is the width of a NN. In modern applications, however at least as much focus if not more lies on the depth of the underlying architectures, which can have more than 1000 layers [HZRS16]. After all, the depth of NNs is responsible for the name of deep learning.

This consideration begs the question of whether there is a concrete mathematically quantifiable benefit of deep architectures over shallow NNs. Indeed, we will see effects of depth at many places throughout this manuscript. However, one of the aspects of deep learning that is most clearly affected by deep architectures is the approximation theoretical aspect. In this framework, we will discuss in Section 3 multiple approaches that describe the effect of depth.

Why do neural networks perform well in very high-dimensional environments? We have seen in Subsection 1.2.2 and will see in Section 3 that from the perspective of approximation theory deep NNs match the performance of the best classical approximation tool in virtually every task. In practice, we observe something that is even more astounding. In fact, NNs seem to perform incredibly well on tasks that no classical, non-specialized approximation method can even remotely handle. The approximation problem that we are talking about here is that of approximation of high-dimensional functions. Indeed, the classical *curse of dimensionality* [Bel52, NW09] postulates that essentially every approximation method deteriorates exponentially fast with increasing dimension.

For example, for the uniform approximation error of 1-Lipschitz continuous functions on a d-dimensional unit cube in the uniform norm, we have a lower bound of $\Omega(p^{-1/d})$, for $p \to \infty$, when approximating with a continuous scheme[17] of p free parameters [DeV98].

On the other hand, in most applications, the input dimensions are massive. For example, the following datasets are typically used as benchmarks in image classification problems: MNIST [LBBH98] with 28×28 pixels per image, CIFAR-10/CIFAR-100 [KH09] with 32×32 pixels per image and ImageNet [DDS$^+$09, KSH12] which contains high-resolution images that are typically down-sampled to 256×256 pixels. Naturally, in real-world applications, the input dimensions may well exceed those of these test problems. However, already for the simplest of the test cases above, the input dimension is $d = 784$. If we use $d = 784$ in the aforementioned lower bound for the approximation of 1-Lipschitz functions, then we require $\mathcal{O}(\varepsilon^{-784})$ parameters to achieve a uniform error of $\varepsilon \in (0, 1)$. Already for moderate ε this value will quickly exceed the storage capacity of any conceivable machine in this universe. Considering the aforementioned curse of dimensionality, it is puzzling to see that NNs perform adequately in this regime. In Section 4, we describe three approaches that offer explanations as to why deep NN-based approximation is not rendered meaningless in the context of high-dimensional input dimensions.

Why does stochastic gradient descent converge to good local minima despite the non-convexity of the problem? As mentioned in Subsection 1.2.1, a convergence guarantee of stochastic gradient descent to a global minimum is typically only given if the underlying objective function admits some form of convexity. However, the empirical risk of a NN, i.e., $\widehat{\mathcal{R}}_s(\Phi(\cdot, \theta))$, is typically not a convex function with respect to the parameters θ. For a simple intuitive reason why this function fails to be convex, it is instructive to consider the following example.

[17] One can achieve better rates at the cost of discontinuous (with respect to the function to be approximated) parameter assignment. This can be motivated by the use of space-filling curves. In the context of NNs with piecewise polynomial activation functions, a rate of $p^{-2/d}$ can be achieved by very deep architectures [Yar18a, YZ20].

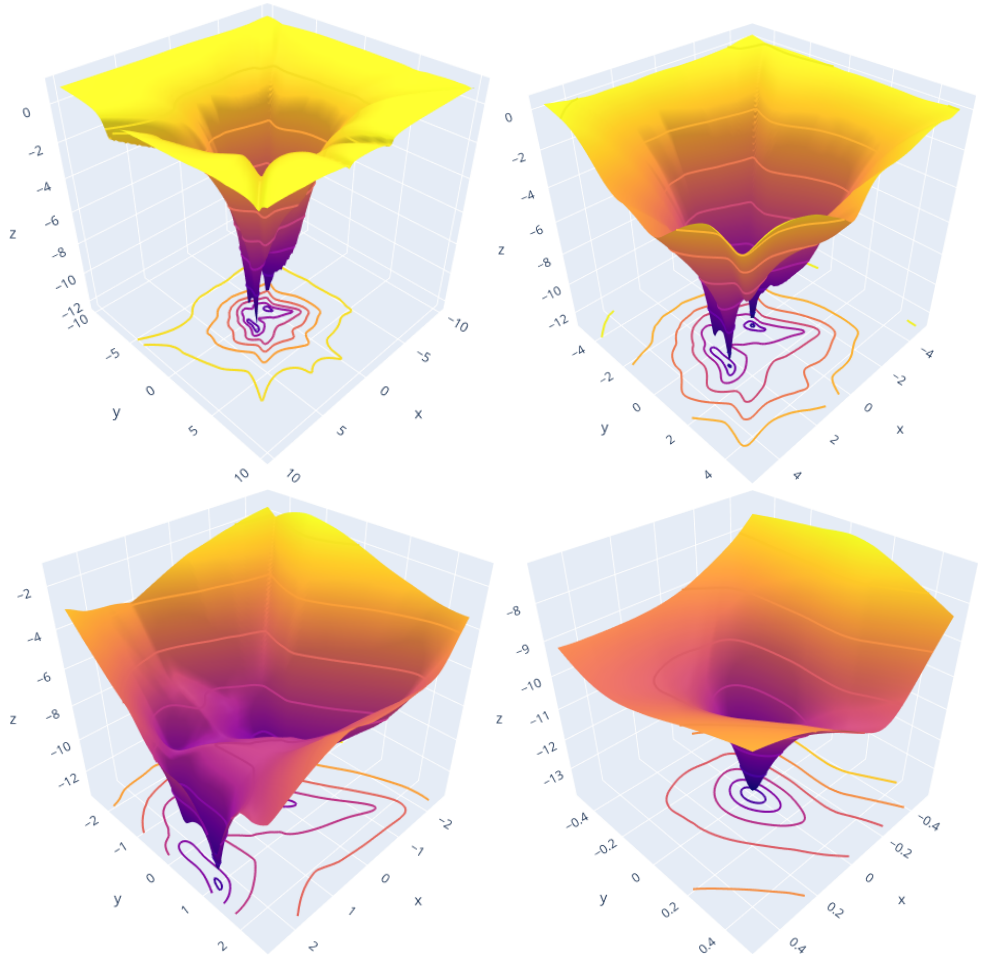


Figure 1.5: Two-dimensional projection of the loss landscape of a neural network with four layers and ReLU activation function on four different scales. From top-left to bottom-right, we zoom into the global minimum of the landscape.

Example 1.20. *Consider the NN*

$$\Phi(x,\theta) = \theta_1 \varrho_R(\theta_3 x + \theta_5) + \theta_2 \varrho_R(\theta_4 x + \theta_6), \qquad \theta \in \mathbb{R}^6, \quad x \in \mathbb{R},$$

with the ReLU activation function $\varrho_R(x) = \max\{0, x\}$*. It is not hard to see that the two parameter values* $\theta = (1, -1, 1, 1, 1, 0)$ *and* $\bar{\theta} = (-1, 1, 1, 1, 0, 1)$ *produce the same realization function*[18]*, i.e.,* $\Phi(\cdot, \theta) = \Phi(\cdot, \bar{\theta})$*. However, since* $(\theta + \bar{\theta})/2 = (0, 0, 1, 1, 1/2, 1/2)$*, we conclude that* $\Phi(\cdot, (\theta + \bar{\theta})/2) = 0$*. Clearly, for the data* $s = ((-1, 0), (1, 1))$*, we now have that*

$$\widehat{\mathcal{R}}_s(\Phi(\cdot, \theta)) = \widehat{\mathcal{R}}_s(\Phi(\cdot, \bar{\theta})) = 0 \quad \textit{and} \quad \widehat{\mathcal{R}}_s\left(\Phi(\cdot, (\theta + \bar{\theta})/2)\right) = \frac{1}{2},$$

showing the non-convexity of $\widehat{\mathcal{R}}_s$*.*

[18] This corresponds to interchanging the two neurons in the hidden layer. In general it holds that the realization function of a FC NN is invariant under permutations of the neurons in a given hidden layer.

Given this non-convexity, Algorithm 1 faces serious challenges. Firstly, there may exist multiple suboptimal local minima. Secondly, the objective may exhibit saddle points, some of which may be of higher order, i.e., the Hessian vanishes. Finally, even if no suboptimal local minima exist, there may be extensive areas of the parameter space where the gradient is very small, so that escaping these regions can take a very long time.

These issues are not mere theoretical possibilities, but will almost certainly arise. For example, [AHW96, SS18] show the existence of many suboptimal local minima in typical learning tasks. Moreover, for fixed-sized NNs, it has been shown in [BEG19, PRV20], that with respect to L^p-norms the set of NNs is generally a very non-convex and non-closed set. Also, the map $\theta \mapsto \Phi_a(\cdot, \theta)$ is not a quotient map, i.e., not continuously invertible when accounting for its non-injectivity. In addition, in various situations finding the global optimum of the minimization problem is shown to be NP-hard in general [BR89, Jud90, Ším02]. In Figure 1.5 we show the two-dimensional projection of a loss landscape, i.e., the projection of the graph of the function $\theta \mapsto \widehat{\mathcal{R}}_s(\Phi(\cdot, \theta))$. It is apparent from the visualization that the problem exhibits more than one minimum. We also want to add that in practice one neglects that the loss is only almost everywhere differentiable in case of piecewise smooth activation functions, such as the ReLU, although one could resort to subgradient methods [KL18].

In view of these considerations, the classical framework presented in Subsection 1.2.1 offers no explanation as to why deep learning works in practice. Indeed, in the survey [OM98, Section 1.4] the state of the art in 1998 was summarized by the following assessment: "There is no formula to guarantee that (1) the NN will converge to a good solution, (2) convergence is swift, or (3) convergence even occurs at all."

Nonetheless, in applications, not only would an explanation of when and why SGD converges be extremely desirable, convergence is also quite often observed even though there is little theoretical explanation for it in the classical set-up. In Section 5, we collect modern approaches explaining why and when convergence occurs and can be guaranteed.

Which aspects of a neural network architecture affect the performance of deep learning? In the introduction to classical approaches to deep learning above, we have seen that in classical results, such as in Theorem 1.16, only the effect of few aspects of the NN architectures are considered. In Theorem 1.16 only the impact of the width of the NN was studied. In further approximation theorems below, e.g., in Theorems 2.1 and 3.2, we will additionally have a variable depth of NNs. However, for deeper architectures, there are many additional aspects of the architecture that could potentially affect the performance of the model for the associated learning task. For example, even for a standard FC NN with L layers as in Definition 1.4, there is a lot of flexibility in choosing the number of neurons $(N_1, \dots, N_{L-1}) \in \mathbb{N}^{L-1}$ in the hidden layers. One would expect that certain choices affect the capabilities of the NNs considerably and some choices are preferable over others. Note that, one aspect of the neural network architecture that can have a profound effect on the performance, especially regarding approximation theoretical aspects of the performance, is the choice of the activation function. For example, in [MP99, Yar21] activation functions were found that allow uniform approximation of continuous functions to arbitrary accuracy with fixed-size neural networks. In the sequel we will, however, focus on architectural aspects other than the activation function.

In addition, practitioners have invented an immense variety of NN architectures for specific problems. These include NNs with convolutional blocks [LBBH98], with skip connections [HZRS16], sparse connections [ZAP16, BBC17], batch normalization blocks [IS15], and many more. In addition, for sequential data, recurrent connections are used [RHW86] and these often have forget mechanisms [HS97] or other gates [CvMG$^+$14] included in their architectures.

The choice of an appropriate NN architecture is essential to the success of many deep learning tasks. This goes so far, that frequently an architecture search is applied to find the most suitable one [ZL17, PGZ$^+$18]. In most cases, though, the design and choice of the architecture is based on the intuition of the practitioner.

Naturally, from a theoretical point of view, this situation is not satisfactory. Instead, it would be highly desirable to have a mathematical theory guiding the choice of NN architectures. More concretely, one would wish for mathematical theorems that identify those architectures that work for a specific problem and those that will yield suboptimal results. In Section 6, we discuss various results that explain theoretically quantifiable effects of certain aspects or building blocks of NN architectures.

Which features of data are learned by deep architectures? It is commonly believed that the neurons of NNs constitute feature extractors in different levels of abstraction that correspond to the layers. This belief is partially grounded in experimental evidence as well as in drawing connections to the human visual cortex, see [GBC16, Chapter 9.10].

Understanding the features that are learned can, in a way, be linked to understanding the reasoning with which a NN-based model ended up with its result. Therefore, analyzing the features that a NN learns constitutes a data-aware approach to understanding deep learning. Naturally, this falls outside of the scope of the classical theory, which is formulated in terms of optimization, generalization, and approximation errors.

One central obstacle towards understanding these features theoretically is that, at least for practical problems, the data distribution is unknown. However, one often has partial knowledge. One example is that in image classification it appears reasonable to assume that any classifier is translation and rotation invariant as well as invariant under small deformations. In this context, it is interesting to understand under which conditions trained NNs admit the same invariances.

Biological NNs such as the visual cortex are believed to be evolved in a way that is based on sparse multiscale representations of visual information [OF96]. Again, a fascinating question is whether NNs trained in practice can be shown to favor such multiscale representations based on sparsity or if the architecture is theoretically linked to sparse representations. We will discuss various approaches studying the features learned by neural networks in Section 7.

Are neural networks capable of replacing highly specialized numerical algorithms in natural sciences? Shortly after their successes in various data-driven tasks in data science and AI applications, NNs have been used also as a numerical ansatz for solving highly complex models from the natural sciences which may be combined with data driven methods. This per se is not very surprising as many such models can be formulated as optimization problems where the common deep learning paradigm can be directly applied. What might be considered surprising is that this approach seems to be applicable to a wide range of problems which have previously been tackled by highly specialized numerical methods.

Particular successes include the data-driven solution of ill-posed *inverse problems* [AMÖS19] which have, for example, led to a fourfold speedup in MRI scantimes [ZKS+18] igniting the research project `fastmri.org`. Deep-learning-based approaches have also been very successful in solving a vast array of different *partial differential equation* (PDE) models, especially in the high-dimensional regime [EY18, RPK19, HSN20, PSMF20] where most other methods would suffer from the curse of dimensionality.

Despite these encouraging applications, the foundational mechanisms governing their workings and limitations are still not well understood. In Subsection 4.3 and Section 8 we discuss some theoretical and practical aspects of deep learning methods applied to the solution of inverse problems and PDEs.

2 Generalization of large neural networks

In the following, we will shed light on the generalization puzzle of NNs as described in Subsection 1.3. We focus on four different lines of research which, of course, do not cover the wide range of available results. In fact, we had to omit a discussion of a multitude of important works, some of which we reference in the following paragraph.

First, let us mention extensions of the generalization bounds presented in Subsection 1.2.3 making use of *local* Rademacher complexities [BBM05] or dropping assumptions on boundedness or rapidly decaying tails [Men14]. Furthermore, there are approaches to generalization which do not focus on the hypothesis set $\mathcal{F}$, i.e., the range of the learning algorithm $\mathcal{A}$, but the way $\mathcal{A}$ chooses its model f_s. For instance, one can assume that f_s does not depend too strongly on each individual sample (*algorithmic stability* [BE02, PRMN04]), only on a subset of the samples (*compression bounds* [AGNZ18]), or satisfies local properties (*algorithmic robustness* [XM12]). Finally, we refer the reader to [JNM+20] and the references mentioned therein for an empirical study of various measures related to generalization.

Note that many results on generalization capabilities of NNs can still only be proven in simplified settings, e.g., for deep linear NNs, i.e., $\varrho(x) = x$, or basic linear models, i.e., one-layer NNs. Thus, we start by

emphasizing the connection of deep, nonlinear NNs to linear models (operating on features given by a suitable kernel) in the *infinite width limit*.

2.1 Kernel regime

We consider a one-dimensional prediction setting where the loss $\mathcal{L}(f,(x,y))$ depends on $x \in \mathcal{X}$ only through $f(x) \in \mathcal{Y}$, i.e., there exists a function $\ell\colon \mathcal{Y}\times\mathcal{Y}\to\mathbb{R}$ such that

$$\mathcal{L}(f,(x,y)) = \ell(f(x),y).$$

For instance, in case of the quadratic loss we have that $\ell(\hat{y},y) = (\hat{y}-y)^2$. Further, let Φ be a NN with architecture $(N,\varrho) = ((d,N_1,\dots,N_{L-1},1),\varrho)$ and let Θ_0 be a $\mathbb{R}^{P(N)}$-valued random variable. For simplicity, we evolve the parameters of Φ according to the continuous version of gradient descent, so-called *gradient flow*, given by

$$\frac{\mathrm{d}\Theta(t)}{\mathrm{d}t} = -\nabla_\theta \widehat{\mathcal{R}}_s(\Phi(\cdot,\Theta(t))) = -\frac{1}{m}\sum_{i=1}^m \nabla_\theta\Phi(x^{(i)},\Theta(t))D_i(t), \quad \Theta(0)=\Theta_0, \tag{2.1}$$

where $D_i(t) \coloneqq \frac{\partial \ell(\hat{y},y^{(i)})}{\partial\hat{y}}|_{\hat{y}=\Phi(x^{(i)},\Theta(t))}$ is the derivative of the loss with respect to the prediction at input feature $x^{(i)}$ at time $t\in[0,\infty)$. The chain rule implies the following dynamics of the NN realization

$$\frac{\mathrm{d}\Phi(\cdot,\Theta(t))}{\mathrm{d}t} = -\frac{1}{m}\sum_{i=1}^m K_{\Theta(t)}(\cdot,x^{(i)})D_i(t) \tag{2.2}$$

and its empirical risk

$$\frac{\mathrm{d}\widehat{\mathcal{R}}_s(\Phi(\cdot,\Theta(t))}{\mathrm{d}t} = -\frac{1}{m^2}\sum_{i=1}^m\sum_{j=1}^m D_i(t)K_{\Theta(t)}(x^{(i)},x^{(j)})D_j(t), \tag{2.3}$$

where K_θ, $\theta\in\mathbb{R}^{P(N)}$, is the so-called *neural tangent kernel* (NTK)

$$K_\theta\colon \mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}, \quad K_\theta(x_1,x_2) = \big(\nabla_\theta\Phi(x_1,\theta)\big)^T\nabla_\theta\Phi(x_2,\theta). \tag{2.4}$$

Now let $\sigma_w,\sigma_b\in(0,\infty)$ and assume that the initialization Θ_0 consists of independent entries, where entries corresponding to the weight matrix and bias vector in the ℓ-th layer follow a normal distribution with zero mean and variances σ_w^2/N_ℓ and σ_b^2, respectively. Under weak assumptions on the activation function, the central limit theorem implies that the pre-activations converge to i.i.d. centered Gaussian processes in the infinite width limit $N_1,\dots,N_{L-1}\to\infty$, see [LBN+18, MHR+18]. Similarly, also K_{Θ_0} converges to a deterministic kernel K^∞ which stays constant in time and only depends on the activation function ϱ, the depth L, and the initialization parameters σ_w and σ_b [JGH18, ADH+19, Yan19, LXS+20]. Thus, within the infinite width limit, gradient flow on the NN parameters as in (2.1) is equivalent to functional gradient flow in the *reproducing kernel Hilbert space* $(\mathcal{H}_{K^\infty},\|\cdot\|_{K^\infty})$ corresponding to K^∞, see (2.2).

By (2.3), the empirical risk converges to a global minimum as long as the kernel evaluated at the input features, $\bar{K}^\infty \coloneqq (K^\infty(x^{(i)},x^{(j)}))_{i,j=1}^m \in \mathbb{R}^{m\times m}$, is positive definite (see, e.g., [JGH18, DLL+19] for suitable conditions) and the $\ell(\cdot,y^{(i)})$ are convex and lower bounded. For instance, in case of the quadratic loss the solution of (2.2) is then given by

$$\Phi(\cdot,\Theta(t)) = C(t)(y^{(i)})_{i=1}^m + \big(\Phi(\cdot,\Theta_0) - C(t)(\Phi(x^{(i)},\Theta_0))_{i=1}^m\big), \tag{2.5}$$

where $C(t) \coloneqq \big((K^\infty(\cdot,x^{(i)}))_{i=1}^m\big)^T(\bar{K}^\infty)^{-1}(\mathrm{I}_m - e^{-\frac{2\bar{K}^\infty t}{m}})$. As the initial realization $\Phi(\cdot,\Theta_0)$ constitutes a centered Gaussian process, the second term in (2.5) follows a normal distribution with zero mean at each input. In the limit $t\to\infty$, its variance vanishes on the input features $x^{(i)}$, $i\in[m]$, and the first term convergences to the minimum kernel-norm interpolator, i.e., to the solution of

$$\min_{f\in\mathcal{H}_{K^\infty}} \|f\|_{K^\infty} \quad \text{s.t.} \quad f(x^{(i)}) = y^{(i)}.$$

Therefore, within the infinite width limit, the generalization properties of the NN could be described by the generalization properties of the minimizer in the reproducing kernel Hilbert space corresponding to the kernel K^∞ [BMM18, LR20, LRZ20, GMMM21, Li21].

This so-called *lazy training*, where a NN essentially behaves like a linear model with respect to the nonlinear features $x \mapsto \nabla_\theta \Phi(x, \theta)$, can already be observed in the non-asymptotic regime, see also Subsection 5.2. For sufficiently overparametrized ($P(N) \gg m$) and suitably initialized models, one can show that $K_{\theta(0)}$ is close to K^∞ at initialization and $K_{\theta(t)}$ stays close to $K_{\theta(0)}$ throughout training, see [DZPS18, ADH+19, COB19, DLL+19]. The dynamics of the NN under gradient flow in (2.2) and (2.3) can thus be approximated by the dynamics of the linearization of Φ at initialization Θ_0, given by

$$\Phi^{\mathrm{lin}}(\cdot, \theta) \coloneqq \Phi(\cdot, \Theta_0) + \langle \nabla_\theta \Phi(\cdot, \Theta_0), \theta - \Theta_0 \rangle,$$

which motivates to study the behavior of linear models in the overparametrized regime.

2.2 Norm-based bounds and margin theory

For piecewise linear activation functions, one can improve upon the VC-dimension bounds in Theorem 1.18 and show that, up to logarithmic factors, the VC-dimension is asymptotically bounded both above and below by $P(N)L$, see [BHLM19]. The lower bound shows that the generalization bound in Theorem 1.19 can only be non-vacuous if the number of samples m scales at least linearly with the number of NN parameters $P(N)$. However, heavily overparametrized NNs used in practice seem to generalize well outside of this regime.

One solution is to bound other complexity measures of NNs taking into account various norms on the parameters and avoid the direct dependence on the number of parameters [Bar98]. For instance, we can compute bounds on the Rademacher complexity of NNs with positively homogeneous activation function, where the Frobenius norm of the weight matrices is bounded, see also [NTS15]. Note that, for instance, the ReLU activation is positively homogeneous, i.e., it satisfies that $\varrho_R(\lambda x) = \lambda \varrho_R(x)$ for all $x \in \mathbb{R}$ and $\lambda \in (0, \infty)$.

Theorem 2.1 (Rademacher complexity of neural networks)**.** *Let $d \in \mathbb{N}$, assume that $\mathcal{X} = B_1(0) \subset \mathbb{R}^d$, and let ϱ be a positively homogeneous activation function with Lipschitz constant 1. We define the set of all biasless NN realizations with depth $L \in \mathbb{N}$, output dimension 1, and Frobenius norm of the weight matrices bounded by $C \in (0, \infty)$ as*

$$\widetilde{\mathcal{F}}_{L,C} \coloneqq \big\{ \Phi_{(N,\varrho)}(\cdot, \theta) \colon N \in \mathbb{N}^{L+1},\ N_0 = d,\ N_L = 1,\ \theta = ((W^{(\ell)}, 0))_{\ell=1}^{L} \in \mathbb{R}^{P(N)},\ \|W^{(\ell)}\|_F \leq C \big\}.$$

Then for every $m \in \mathbb{N}$ it holds that

$$\mathfrak{R}_m(\widetilde{\mathcal{F}}_{L,C}) \leq \frac{C(2C)^{L-1}}{\sqrt{m}}.$$

The term 2^{L-1} depending exponentially on the depth can be reduced to $\sqrt{L}$ or completely omitted by invoking also the spectral norm of the weight matrices [GRS18]. Further, observe that for $L = 1$, i.e., linear classifiers with bounded Euclidean norm, this bound is independent of the input dimension d. Together with (1.12), this motivates why the regularized linear model in Figure 1.4 did perform well in the overparametrized regime.

The proof of Theorem 2.1 is based on the contraction property of the Rademacher complexity [LT91] which establishes that

$$\mathfrak{R}_m(\varrho \circ \widetilde{\mathcal{F}}_{\ell,C}) \leq 2\mathfrak{R}_m(\widetilde{\mathcal{F}}_{\ell,C}), \quad \ell \in \mathbb{N}.$$

We can iterate this together with the fact that for every $\tau \in \{-1, 1\}^m$, and $x \in \mathbb{R}^{N_{\ell-1}}$ it holds that

$$\sup_{\|W^{(\ell)}\|_F \leq C} \Big\| \sum_{i=1}^{m} \tau_i \varrho(W^{(\ell)} x) \Big\|_2 = C \sup_{\|w\|_2 \leq 1} \Big| \sum_{i=1}^{m} \tau_i \varrho(\langle w, x \rangle) \Big|.$$

In summary, one establishes that

$$\mathfrak{R}_m(\widetilde{\mathcal{F}}_{L,C}) = \frac{C}{m}\mathbb{E}\big[\sup_{f\in\widetilde{\mathcal{F}}_{L-1,C}} \big\|\sum_{i=1}^m \tau_i \varrho(f(X^{(i)}))\big\|_2\big] \leq \frac{C(2C)^{L-1}}{m}\mathbb{E}\big[\big\|\sum_{i=1}^m \tau_i X^{(i)}\big\|_2\big],$$

which by Jensen's inequality yields the claim.

Recall that for classification problems one typically minimizes a surrogate loss $\mathcal{L}^{\mathrm{surr}}$, see Remark 1.9. This suggests that there could be a trade-off happening between complexity of the hypothesis class $\mathcal{F}_a$ and the corresponding regression fit underneath, i.e., the *margin* $M(f,z) \coloneqq yf(x)$ by which a training example $z=(x,y)$ has been classified correctly by $f\in\mathcal{F}_a$, see [BFT17, NBS18, JKMB19]. For simplicity, let us focus on the ramp surrogate loss with confidence $\gamma>0$, i.e., $\mathcal{L}^{\mathrm{surr}}_\gamma(f,z) \coloneqq \ell_\gamma(M(f,z))$, where

$$\ell_\gamma(t) \coloneqq \mathbb{1}_{(-\infty,\gamma]}(t) - \tfrac{t}{\gamma}\mathbb{1}_{[0,\gamma]}(t), \quad t\in\mathbb{R}.$$

Note that the ramp function ℓ_γ is $1/\gamma$-Lipschitz continuous. Using McDiarmid's inequality and a symmetrization argument similar to the proof of Theorem 1.19, combined with the contraction property of the Rademacher complexity, yields the following bound on the probability of misclassification: With probability $1-\delta$ for every $f\in\mathcal{F}_a$ it holds that

$$\begin{aligned}
\mathbb{P}[\operatorname{sgn}(f(X))\neq Y] \leq \mathbb{E}\big[\mathcal{L}^{\mathrm{surr}}_\gamma(f,Z)\big] &\lesssim \frac{1}{m}\sum_{i=1}^m \mathcal{L}^{\mathrm{surr}}_\gamma(f,Z^{(i)}) + \mathfrak{R}_m(\mathcal{L}^{\mathrm{surr}}_\gamma\circ\mathcal{F}_a) + \sqrt{\frac{\ln(1/\delta)}{m}} \\
&\lesssim \frac{1}{m}\sum_{i=1}^m \mathbb{1}_{(-\infty,\gamma)}(Y^{(i)}f(X^{(i)})) + \frac{\mathfrak{R}_m(M\circ\mathcal{F}_a)}{\gamma} + \sqrt{\frac{\ln(1/\delta)}{m}} \\
&= \frac{1}{m}\sum_{i=1}^m \mathbb{1}_{(-\infty,\gamma)}(Y^{(i)}f(X^{(i)})) + \frac{\mathfrak{R}_m(\mathcal{F}_a)}{\gamma} + \sqrt{\frac{\ln(1/\delta)}{m}}.
\end{aligned}$$

This shows the trade-off between the complexity of $\mathcal{F}_a$ measured by $\mathfrak{R}_m(\mathcal{F}_a)$ and the fraction of training data that has been classified correctly with a margin of at least γ. In particular this suggests, that (even if we classify the training data correctly with respect to the 0-1 loss) it might be beneficial to further increase the complexity of $\mathcal{F}_a$ to simultaneously increase the margins by which the training data has been classified correctly and thus obtain a better generalization bound.

2.3 Optimization and implicit regularization

The optimization algorithm, which is usually a variant of SGD, seems to play an important role for the generalization performance. Potential indicators for good generalization performance are high speed of convergence [HRS16] or flatness of the local minimum to which SGD converged, which can be characterized by the magnitude of the eigenvalues of the Hessian (or approximately as the robustness of the minimizer to adversarial perturbations on the parameter space), see [KMN+17]. In [DR17, NBMS17] generalization bounds depending on a concept of flatness are established by employing a PAC-Bayesian framework, which can be viewed as a generalization of Theorem 1.17, see [McA99]. Further, one can also unite flatness and norm-based bounds by the *Fisher–Rao metric* of information geometry [LPRS19].

Let us motivate the link between generalization and flatness in the case of simple linear models: We assume that our model takes the form $\langle\theta,\cdot\rangle$, $\theta\in\mathbb{R}^d$, and we will use the abbreviations

$$r(\theta) \coloneqq \widehat{\mathcal{R}}_s(\langle\theta,\cdot\rangle) \quad\text{and}\quad \gamma(\theta) \coloneqq \min_{i\in[m]} M(\langle\theta,\cdot\rangle, z^{(i)}) = \min_{i\in[m]} y^{(i)}\langle\theta,x^{(i)}\rangle$$

throughout this subsection to denote the empirical risk and the margin for given training data $s = ((x^{(i)},y^{(i)}))_{i=1}^m$. We assume that we are solving a classification task with the 0-1 loss and that our training

data is linearly separable. This means that there exists a minimizer $\hat{\theta} \in \mathbb{R}^d$ such that $r(\hat{\theta}) = 0$. We observe that δ-robustness in the sense that

$$\max_{\theta \in B_\delta(0)} r(\hat{\theta} + \theta) = r(\hat{\theta}) = 0$$

implies that

$$0 < \min_{i \in [m]} y^{(i)} \langle \hat{\theta} - \delta y^{(i)} \tfrac{x^{(i)}}{\|x^{(i)}\|_2}, x^{(i)} \rangle \leq \gamma(\hat{\theta}) - \delta \min_{i \in [m]} \|x^{(i)}\|_2,$$

see also [PKL+17]. This lower bound on the margin $\gamma(\hat{\theta})$ then ensures generalization guarantees as described in Subsection 2.2.

Even without explicit[19] control on the complexity of $\mathcal{F}_a$, there do exist results showing that SGD acts as implicit regularization [NTS14]. This is motivated by linear models where SGD converges to the minimal Euclidean norm solution for the quadratic loss and in the direction of the hard margin support vector machine solution for the logistic loss on linearly separable data [SHN+18]. Note that convergence to minimum norm or maximum margin solutions in particular decreases the complexity of our hypothesis set and thus improves generalization bounds, see Subsection 2.2.

While we have seen this behavior of gradient descent for linear regression already in the more general context of kernel regression in Subsection 2.1, we want to motivate the corresponding result for classification tasks in the following. We focus on the exponential surrogate loss $\mathcal{L}^{\text{surr}}(f, z) = \ell(M(f, z)) = e^{-yf(x)}$ with $\ell(z) = e^{-z}$, but similar observations can be made for the logistic loss defined in Remark 1.9. We assume that the training data is linearly separable, which guarantees the existence of $\hat{\theta} \neq 0$ with $\gamma(\hat{\theta}) > 0$. Then for every linear model $\langle \theta, \cdot \rangle$, $\theta \in \mathbb{R}^d$, it holds that

$$\left\langle \hat{\theta}, \nabla_\theta r(\theta) \right\rangle = \frac{1}{m} \sum_{i=1}^{m} \underbrace{\ell'(y^{(i)} \langle \theta, x^{(i)} \rangle)}_{<0} \underbrace{y^{(i)} \langle \hat{\theta}, x^{(i)} \rangle}_{>0}.$$

A critical point $\nabla_\theta r(\theta) = 0$ can therefore be approached if and only if for every $i \in [m]$ we have

$$\ell'(y^{(i)} \langle \theta, x^{(i)} \rangle) = -e^{-y^{(i)} \langle \theta, x^{(i)} \rangle} \to 0,$$

which is equivalent to $\|\theta\|_2 \to \infty$ and $\gamma(\theta) > 0$. Let us now define

$$r_\beta(\theta) \coloneqq \frac{\ell^{-1}(r(\beta\theta))}{\beta}, \quad \theta \in \mathbb{R}^d, \ \beta \in (0, \infty),$$

and observe that

$$r_\beta(\theta) = -\frac{\log(r(\beta\theta))}{\beta} \to \gamma(\theta), \quad \beta \to \infty. \tag{2.6}$$

Due to this property, r_β is often referred to as the *smoothed margin* [LL19, JT19b]. We evolve θ according to gradient flow with respect to the smoothed margin r_1, i.e.,

$$\frac{\mathrm{d}\theta(t)}{\mathrm{d}t} = \nabla_\theta r_1(\theta(t)) = -\frac{1}{r(\theta(t))} \nabla_\theta r(\theta(t)),$$

which produces the same trajectory as gradient flow with respect to the empirical risk r under a rescaling of the time t. Looking at the evolution of the normalized parameters $\tilde{\theta}(t) = \theta(t)/\|\theta(t)\|_2$, the chain rule establishes that

$$\frac{\mathrm{d}\tilde{\theta}(t)}{\mathrm{d}t} = P_{\tilde{\theta}(t)} \frac{\nabla_\theta r_{\beta(t)}(\tilde{\theta}(t))}{\beta(t)} \quad \text{with} \quad \beta(t) \coloneqq \|\theta(t)\|_2 \quad \text{and} \quad P_\theta \coloneqq \mathrm{I}_d - \theta\theta^T, \quad \theta \in \mathbb{R}^d.$$

[19] Note that also different architectures can exhibit vastly different inductive biases [ZBH+20] and also within the architecture different parameters have different importance, see [FC18, ZBS19] and Proposition 6.2.

This shows that the normalized parameters perform projected gradient ascent with respect to the function $r_{\beta(t)}$, which converges to the margin due to (2.6) and the fact that $\beta(t) = \|\theta(t)\|_2 \to \infty$ when approaching a critical point. This motivates that during gradient flow the normalized parameters implicitly maximize the margin. See [GLSS18a, GLSS18b, LL19, NLG+19, CB20, JT20] for a precise analysis and various extensions, e.g., to homogeneous or two-layer NNs and other optimization geometries.

To illustrate one research direction, we present an exemplary result in the following. Let $\Phi = \Phi_{(N,\varrho)}$ be a biasless NN with parameters $\theta = ((W^{(\ell)}, 0))_{\ell=0}^{L}$ and output dimension $N_L = 1$. For given input features $x \in \mathbb{R}^{N_0}$, the gradient $\nabla_{W^{(\ell)}}\Phi = \nabla_{W^{(\ell)}}\Phi(x,\theta) \in \mathbb{R}^{N_{\ell-1}\times N_\ell}$ with respect to the weight matrix in the ℓ-th layer satisfies that

$$\nabla_{W^{(\ell)}}\Phi = \varrho(\Phi^{(\ell-1)})\frac{\partial\Phi}{\partial\Phi^{(\ell+1)}}\frac{\partial\Phi^{(\ell+1)}}{\partial\Phi^{(\ell)}} = \varrho(\Phi^{(\ell-1)})\frac{\partial\Phi}{\partial\Phi^{(\ell+1)}}W^{(\ell+1)}\operatorname{diag}\big(\varrho'(\Phi^{(\ell)})\big),$$

where the pre-activations $(\Phi^{(\ell)})_{\ell=1}^{L}$ are given as in (1.1). Evolving the parameters according to gradient flow as in (2.1) and using an activation function ϱ with $\varrho(x) = \varrho'(x)x$, such as the ReLU, this implies that

$$\operatorname{diag}\big(\varrho'(\Phi^{(\ell)})\big)W^{(\ell)}(t)\Big(\frac{\mathrm{d}W^{(\ell)}(t)}{\mathrm{d}t}\Big)^T = \Big(\frac{\mathrm{d}W^{(\ell+1)}(t)}{\mathrm{d}t}\Big)^T W^{(\ell+1)}(t)\operatorname{diag}\big(\varrho'(\Phi^{(\ell)})\big). \tag{2.7}$$

Note that this ensures the conservation of balancedness between weight matrices of adjacent layers, i.e.,

$$\frac{\mathrm{d}}{\mathrm{d}t}\big(\|W^{(\ell+1)}(t)\|_F^2 - \|W^{(\ell)}(t)\|_F^2\big) = 0,$$

see [DHL18]. Furthermore, for deep linear NNs, i.e., $\varrho(x) = x$, the property in (2.7) implies conservation of alignment of left and right singular spaces of $W^{(\ell)}$ and $W^{(\ell+1)}$. This can then be used to show implicit preconditioning and convergence of gradient descent [ACH18, ACGH19] and that, under additional assumptions, gradient descent converges to a linear predictor that is aligned with the maximum margin solution [JT19a].

2.4 Limits of classical theory and double descent

There is ample evidence that classical tools from statistical learning theory alone, such as Rademacher averages, uniform convergence, or algorithmic stability may be unable to explain the full generalization capabilities of NNs [ZBH+17, NK19]. It is especially hard to reconcile the classical bias-variance trade-off with the observation of good generalization performance when achieving zero empirical risk on noisy data using a regression loss. On top of that, this behavior of overparametrized models in the interpolation regime turns out not to be unique to NNs. Empirically, one observes for various methods (decision trees, random features, linear models) that the test error decreases even below the sweet-spot in the u-shaped bias-variance curve when further increasing the number of parameters [BHMM19, GJS+20, NKB+20]. This is often referred to as the *double descent curve* or *benign overfitting*, see Figure 2.1. For special cases, e.g., linear regression or random feature regression, such behavior can even be proven, see [HMRT19, MM19, BLLT20, BHX20, MVSS20].

In the following we analyze this phenomenon in the context of linear regression. Specifically, we focus on a prediction task with quadratic loss, input features given by a centered $\mathbb{R}^d$-valued random variable X, and labels given by $Y = \langle\theta^*, X\rangle + \nu$, where $\theta^* \in \mathbb{R}^d$ and ν is a centered random variable independent of X. For training data $S = ((X^{(i)}, Y^{(i)}))_{i=1}^m$, we consider the empirical risk minimizer $\widehat{f}_S = \langle\hat{\theta}, \cdot\rangle$ with minimum Euclidean norm of its parameters $\hat{\theta}$ or, equivalently, the limit of gradient flow with zero initialization. Using (1.3) and a bias-variance decomposition we can write

$$\begin{aligned}\mathbb{E}[\mathcal{R}(\widehat{f}_S)|(X^{(i)})_{i=1}^m] - \mathcal{R}^* &= \mathbb{E}[\|\widehat{f}_S - f^*\|_{L^2(\mathbb{P}_X)}|(X^{(i)})_{i=1}^m] \\ &= (\theta^*)^T P\mathbb{E}[XX^T]P\theta^* + \mathbb{E}[\nu^2]\operatorname{Tr}\big(\Sigma^+\mathbb{E}[XX^T]\big),\end{aligned}$$

where $\Sigma \coloneqq \sum_{i=1}^m X^{(i)}(X^{(i)})^T$, Σ^+ denotes the Moore–Penrose inverse of Σ, and $P \coloneqq \mathrm{I}_d - \Sigma^+\Sigma$ is the orthogonal projector onto the kernel of Σ. For simplicity, we focus on the variance $\operatorname{Tr}\big(\Sigma^+\mathbb{E}[XX^T]\big)$, which can

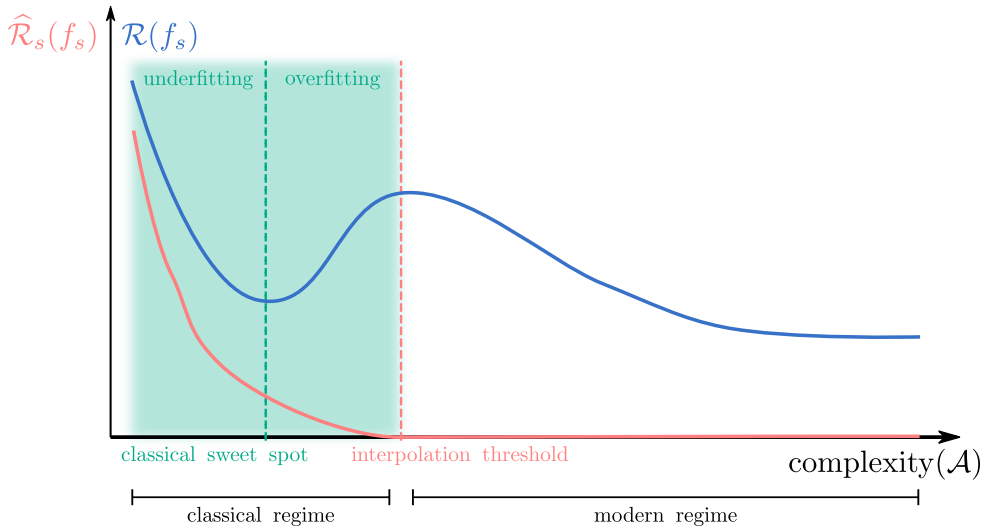


Figure 2.1: This illustration shows the classical, underparametrized regime in green, where the u-shaped curve depicts the bias-variance trade-off as explained in Section 1.2. Starting with complexity of our algorithm $\mathcal{A}$ larger than the interpolation threshold we can achieve zero empirical risk $\widehat{\mathcal{R}}_s(f_s)$ (training error), where $f_s = \mathcal{A}(s)$. Within this modern interpolation regime, the risk $\mathcal{R}(f_s)$ (test error) might be even lower than at the classical sweet spot. Whereas complexity($\mathcal{A}$) traditionally refers to the complexity of the hypothesis set $\mathcal{F}$, there is evidence that also the optimization scheme and the data is influencing the complexity leading to definitions like complexity($\mathcal{A}$) $:= \max\{m \in \mathbb{N}\colon \mathbb{E}[\widehat{\mathcal{R}}_S(\mathcal{A}(S))] \leq \varepsilon \text{ with } S \sim \mathbb{P}_Z^m\}$, for suitable $\varepsilon > 0$ [NKB+20]. This illustration is based on [BHMM19].

be viewed as setting $\theta^* = 0$ and $\mathbb{E}[\nu^2] = 1$. Assuming that X has i.i.d. entries with unit variance and bounded fifth moment, the distribution of the eigenvalues of $\frac{1}{m}\Sigma^+$ in the limit $d, m \to \infty$ with $\frac{d}{m} \to \kappa \in (0, \infty)$ can be described via the Marchenko–Pastur law. Therefore, the asymptotic variance can be computed explicitly as

$$\operatorname{Tr}\big(\Sigma^+ \mathbb{E}[XX^T]\big) \to \frac{1 - \max\{1-\kappa, 0\}}{|1-\kappa|} \quad \text{for} \quad d, m \to \infty \quad \text{with} \quad \frac{d}{m} \to \kappa,$$

almost surely, see [HMRT19]. This shows that despite interpolating the data we can decrease the risk in the overparametrized regime $\kappa > 1$. In the limit $d, m \to \infty$, such benign overfitting can also be shown for more general settings (including lazy training of NNs), some of which even achieve their optimal risk in the overparametrized regime [MM19, MZ20, LD21].

For normally distributed input features X such that $\mathbb{E}[XX^T]$ has rank larger than m, one can also compute the behavior of the variance in the non-asymptomatic regime [BLLT20]. Define

$$k^* := \min\{k \geq 0\colon \frac{\sum_{i>k} \lambda_i}{\lambda_{k+1}} \geq cm\},$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$ are the eigenvalues of $\mathbb{E}[XX^T]$ in decreasing order and $c \in (0, \infty)$ is a universal constant. Assuming that k^*/m is sufficiently small, with high probability it holds that

$$\operatorname{Tr}\big(\Sigma^+ \mathbb{E}[XX^T]\big) \approx \frac{k^*}{m} + \frac{m \sum_{i>k^*} \lambda_i^2}{(\sum_{i>k^*} \lambda_i)^2}.$$

This precisely characterizes the regimes for benign overfitting in terms of the eigenvalues of the covariance matrix $\mathbb{E}[XX^T]$. Furthermore, it shows that adding new input feature coordinates and thus increasing the number of parameters d can lead to either an increase or decrease of the risk.

To motivate this phenomenon, which is considered in much more depth in [CMBK20], let us focus on a single sample $m = 1$ and features X that take values in $\mathcal{X} = \{-1, 1\}^d$. Then it holds that $\Sigma^+ = \frac{X^{(1)}(X^{(1)})^T}{\|X^{(1)}\|^4} = \frac{X^{(1)}(X^{(1)})^T}{d^2}$ and thus

$$\mathbb{E}\big[\mathrm{Tr}\big(\Sigma^+ \mathbb{E}[XX^T]\big)\big] = \frac{1}{d^2}\big\|\mathbb{E}\big[XX^T\big]\big\|_F^2. \tag{2.8}$$

In particular, this shows that incrementing the input feature dimensions $d \mapsto d+1$ one can increase or decrease the risk depending on the correlation of the coordinate X_{d+1} with respect to the previous coordinates $(X_i)_{i=1}^d$, see also Figure 2.2.

Generally speaking, overparametrization and perfectly fitting noisy data does not exclude good generalization performance, see also [BRT19]. However, the risk crucially depends on the data distribution and the chosen algorithm.

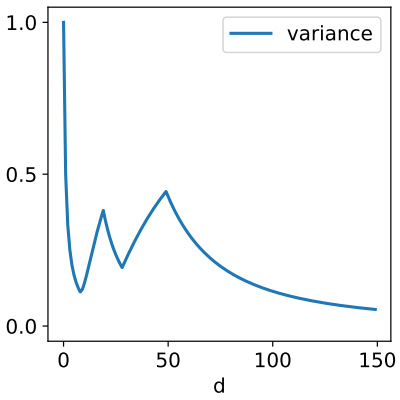


Figure 2.2: The expected variance of linear regression in (2.8) with $d \in [150]$ and $X_i \sim U(\{-1, 1\})$, $i \in [150]$, where $X_i = X_1$ for $i \in \{10, \dots, 20\} \cup \{30, \dots, 50\}$ and all other coordinates are independent.

3 The role of depth in the expressivity of neural networks

The approximation theoretical aspect of a NN architecture, responsible for the approximation component $\varepsilon^{\text{approx}} := \mathcal{R}(f_{\mathcal{F}}^*) - \mathcal{R}^*$ of the error $\mathcal{R}(f_S) - \mathcal{R}^*$ in (1.4), is probably one of the most well-studied parts of the deep learning pipe-line. The achievable approximation error of an architecture most directly describes the power of the architecture.

As mentioned in Subsection 1.3, many classical approaches only study the approximation theory of NNs with few layers, whereas modern architectures are typically very deep. A first observation into the effect of depth is that it can often compensate for insufficient width. For example, in the context of the universal approximation theorem, it was shown that very narrow NNs are still universal if instead of increasing the width, the number of layers can be chosen arbitrarily [HS17, Han19, KL20]. However, if the width of a NN falls below a critical number, then the universality will not hold any longer.

Below, we discuss three additional observations that shed light on the effect of depth on the approximation capacities or alternative notions of expressivity of NNs.

3.1 Approximation of radial functions

One technique to study the impact of depth relies on the construction of specific functions which can be well approximated by NNs of a certain depth, but require significantly more parameters when approximated to the same accuracy by NNs of smaller depth. In the following we present one example for this type of approach, which can be found in [ES16].

Theorem 3.1 (Power of depth). *Let $\varrho \in \{\varrho_R, \varrho_\sigma, \mathbb{1}_{(0,\infty)}\}$ be the ReLU, the logistic, or the Heaviside function. Then there exist constants $c, C \in (0, \infty)$ with the following property: For every $d \in \mathbb{N}$ with $d \geq C$ there exist a probability measure μ on $\mathbb{R}^d$, a three-layer NN architecture $a = (N, \varrho) = ((d, N_1, N_2, 1), \varrho)$ with $\|N\|_\infty \leq Cd^5$, and corresponding parameters $\theta^* \in \mathbb{R}^{P(N)}$ with $\|\theta^*\|_\infty \leq Cd^C$ and $\|\Phi_a(\cdot, \theta^*)\|_{L^\infty(\mathbb{R}^d)} \leq 2$ such that for every $n \leq ce^{cd}$ it holds that*

$$\inf_{\theta \in \mathbb{R}^{P((d,n,1))}} \|\Phi_{((d,n,1),\varrho)}(\cdot, \theta) - \Phi_a(\cdot, \theta^*)\|_{L^2(\mu)} \geq c.$$

In fact, the activation function in Theorem 3.1 is only required to satisfy mild conditions and the result holds, for instance, also for more general sigmoidal functions. The proof of Theorem 3.1 is based on the construction of a suitable radial function $g\colon \mathbb{R}^d \to \mathbb{R}$, i.e., $g(x) = \tilde{g}(\|x\|_2^2)$ for some $\tilde{g}\colon [0,\infty) \to \mathbb{R}$, which can be efficiently approximated by three-layer NNs but approximation by only a two-layer NN requires exponentially large complexity, i.e., the width being exponential in d.

The first observation of [ES16] is that g can typically be well approximated on a bounded domain by a three-layer NN, if $\tilde{g}$ is Lipschitz continuous. Indeed, for the ReLU activation function it is not difficult to show that, emulating a linear interpolation, one can approximate a univariate C-Lipschitz function uniformly on $[0,1]$ up to precision ε by a two-layer architecture of width $\mathcal{O}(C/\varepsilon)$. The same holds for smooth, non-polynomial activation functions due to Theorem 1.16. This implies that the squared Euclidean norm, as a sum of d univariate functions, i.e., $[0,1]^d \ni x \mapsto \sum_{i=1}^d x_i^2$, can be approximated up to precision ε by a two-layer architecture of width $\mathcal{O}(d^2/\varepsilon)$. Moreover, this shows that the third layer can efficiently approximate $\tilde{g}$, establishing approximation of g on a bounded domain up to precision ε using a three-layer architecture with number of parameters polynomial in d/ε.

The second step in [ES16] is to choose g in such a way that the realization of any two-layer neural network $\Phi = \Phi_{((d,n,1),\varrho)}(\cdot,\theta)$ with width n not being exponential in d is on average (with respect to the probability measure μ) a constant distance away from g. Their argument is heavily based on ideas from Fourier analysis and will be outlined below. In this context, let us recall that we denote by $\hat{f}$ the Fourier transform of a suitable function or, more generally, tempered distribution f.

Assuming that the square-root φ of the density function associated with the probability measure μ as well as Φ and g are well-behaved, the Plancherel theorem yields that

$$\|\Phi - g\|_{L^2(\mu)}^2 = \|\Phi\varphi - g\varphi\|_{L^2(\mathbb{R}^d)}^2 = \big\|\widehat{\Phi\varphi} - \widehat{g\varphi}\big\|_{L^2(\mathbb{R}^d)}^2. \tag{3.1}$$

Next, the specific structure of two-layer NNs is used, which implies that for every $j \in [n]$ there exists $w_j \in \mathbb{R}^d$ with $\|w_j\|_2 = 1$ and $\varrho_j\colon \mathbb{R} \to \mathbb{R}$ (subsuming the activation function ϱ, the norm of w_j, and the remaining parameters corresponding to the j-th neuron in the hidden layer) such that Φ is of the form

$$\Phi = \sum_{j=1}^n \varrho_j(\langle w_j, \cdot\rangle) = \sum_{j=1}^n (\varrho_j \otimes \mathbb{1}_{\mathbb{R}^{d-1}}) \circ R_{w_j}.$$

The second equality follows by viewing the action of the j-th neuron as a tensor product of ϱ_j and the indicator function $\mathbb{1}_{\mathbb{R}^{d-1}}(x) = 1$, $x \in \mathbb{R}^{d-1}$, composed with a d-dimensional rotation $R_{w_j} \in SO(d)$ which maps w_j to the first standard basis vector $e^{(1)} \in \mathbb{R}^d$. Noting that the Fourier transform respects linearity, rotations, and tensor products, we can compute

$$\hat{\Phi} = \sum_{j=1}^n (\hat{\varrho}_j \otimes \delta_{\mathbb{R}^{d-1}}) \circ R_{w_j},$$

where $\delta_{\mathbb{R}^{d-1}}$ denotes the Dirac distribution on $\mathbb{R}^{d-1}$. In particular, the support of $\hat{\Phi}$ has a particular star-like shape, namely $\bigcup_{j=1}^n \operatorname{span}\{w_j\}$, which are in fact lines passing through the origin.

Now we choose φ to be the inverse Fourier transform of the indicator function of a ball $B_r(0) \subset \mathbb{R}^d$ with $\operatorname{vol}(B_r(0)) = 1$, ensuring that φ^2 is a valid probability density for μ as

$$\mu(\mathbb{R}^d) = \|\varphi^2\|_{L^1(\mathbb{R}^d)} = \|\varphi\|_{L^2(\mathbb{R}^d)}^2 = \|\hat{\varphi}\|_{L^2(\mathbb{R}^d)}^2 = \|\mathbb{1}_{B_r(0)}\|_{L^2(\mathbb{R}^d)}^2 = 1.$$

Using the convolution theorem, this choice of φ yields that

$$\operatorname{supp}(\widehat{\Phi\varphi}) = \operatorname{supp}(\hat{\Phi} * \hat{\varphi}) \subset \bigcup_{j=1}^n \left(\operatorname{span}\{w_j\} + B_r(0)\right).$$

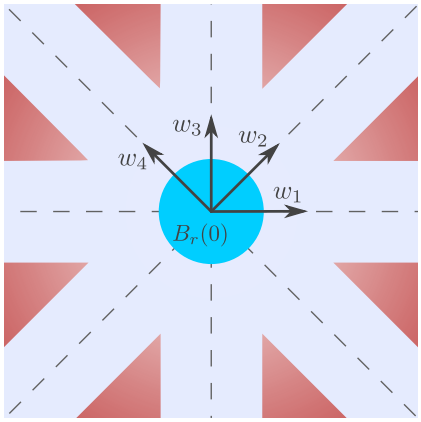


Figure 3.1: This illustration shows the largest possible support (blue) of $\widehat{\Phi\varphi}$, where $\hat{\varphi} = \mathbb{1}_{B_r(0)}$ and Φ is a shallow neural network with architecture $N = (2,4,1)$ and weight matrix $W^{(1)} = [w_1 \dots w_4]^T$ in the first layer. Any radial function with enough of its L^2-mass located at high frequencies (indicated by the red area) cannot be well approximated by $\Phi\varphi$.

Thus the lines passing through the origin are enlarged to tubes. It is this particular shape which allows the construction of some g so that $\|\widehat{\Phi\varphi} - \widehat{g\varphi}\|^2_{L^2(\mathbb{R}^d)}$ can be suitably lower bounded, see also Figure 3.1. Intriguingly, the peculiar behavior of high-dimensional sets now comes into play. Due to the well known concentration of measure principle, the variable n needs to be exponentially large for the set $\bigcup_{j=1}^n (\operatorname{span}\{w_j\} + B_r(0))$ to be not sparse. If it is smaller, one can construct a function g so that the main energy content of $\widehat{g\varphi}$ has a certain distance from the origin, yielding a lower bound for $\|\widehat{\Phi\varphi} - \widehat{g\varphi}\|^2$ and hence $\|\Phi - g\|^2_{L^2(\mu)}$, see (3.1). One key technical problem is the fact that such a behavior for $\hat{g}$ does not immediately imply a similar behavior of $\widehat{g\varphi}$, requiring a quite delicate construction of g.

3.2 Deep ReLU networks

Maybe for no activation function is the effect of depth clearer than for the ReLU activation function $\varrho_R(x) = \max\{0, x\}$. We refer to corresponding NN architectures (N, ϱ_R) as *ReLU (neural) networks* (ReLU NNs). A two-layer ReLU NN with one-dimensional input and output is a function of the form

$$\Phi(x) = \sum_{i=1}^n w_i^{(2)} \varrho_R(w_i^{(1)} x + b_i^{(1)}) + b^{(2)}, \quad x \in \mathbb{R},$$

where $w_i^{(1)}, w_i^{(2)}, b_i^{(1)}, b^{(2)} \in \mathbb{R}$ for $i \in [n]$. It is not hard to see that Φ is a continuous piecewise affine linear function. Moreover, Φ has at most $n+1$ affine linear pieces. On the other hand, notice that the *hat function*

$$h\colon [0,1] \to [0,1],$$
$$x \mapsto 2\varrho_R(x) - 4\varrho_R(x - \tfrac{1}{2}) = \begin{cases} 2x, & \text{if } 0 \le x < \frac{1}{2}, \\ 2(1-x), & \text{if } \frac{1}{2} \le x \le 1, \end{cases}$$

is a NN with two layers and two neurons. Telgarsky observed that the n-fold convolution $h_n(x) := h \circ \cdots \circ h$ produces a sawtooth function with 2^n spikes [Tel15]. In particular, h_n admits 2^n affine linear pieces with only $2n$ many neurons. In this case, we see that deep ReLU NNs are in some sense exponentially more efficient in generating affine linear pieces.

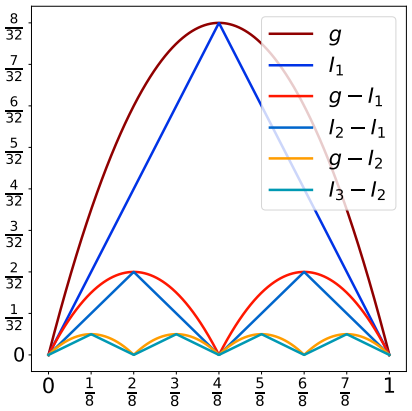


Figure 3.2: Interpolation I_n of $[0,1] \ni x \mapsto g(x) := x - x^2$ on $2^n + 1$ equidistant points, which can be represented as a sum $I_n = \sum_{k=1}^n I_k - I_{k-1} = \sum_{k=1}^n \frac{h_k}{2^{2k}}$ of n sawtooth functions. Each sawtooth function $h_k = h_{k-1} \circ h$ in turn can be written as a k-fold composition of a hat function h. This illustration is based on [EPGB19].

Moreover, it was noted in [Yar17] that the difference of interpolations of $[0,1] \ni x \mapsto x - x^2$ at $2^n + 1$ and $2^{n-1} + 1$ equidistant points equals the scaled sawtooth function $\frac{h_n}{2^{2n}}$, see Figure 3.2. This allows to efficiently implement approximative squaring and, by polarization, also approximative multiplication using ReLU NNs. Composing these simple functions one can approximate localized Taylor polynomials and thus smooth functions, see [Yar17]. We state below a generalization [GKP20] of the result of [Yar17] which includes more general norms, but for $p = \infty$ and $s = 0$ coincides with the original result of Dmitry Yarotsky.

Theorem 3.2 (Approximation of Sobolev-regular functions)**.** *Let $d, k \in \mathbb{N}$ with $k \ge 2$, let $p \in [1, \infty]$, $s \in [0, 1]$, $B \in (0, \infty)$, and let ϱ be a piecewise linear activation function with at least one break-point. Then there exists a constant $c \in (0, \infty)$ with the following property: For every $\varepsilon \in (0, 1/2)$ there exists a NN architecture $a = (N, \varrho)$ with*

$$P(N) \le c\varepsilon^{-d/(k-s)} \log(1/\varepsilon)$$

such that for every function $g \in W^{k,p}((0,1)^d)$ with $\|g\|_{W^{k,p}((0,1)^d)} \le B$ it holds that

$$\inf_{\theta \in \mathbb{R}^{P(N)}} \|\Phi_a(\theta, \cdot) - g\|_{W^{s,p}((0,1)^d)} \le \varepsilon.$$

The ability of deep ReLU neural networks to emulate multiplication has also been employed to reapproximate wide ranges of high-order finite element spaces. In [OPS20] and [MOPS20] it was shown that deep ReLU neural networks are capable of achieving the approximation rates of *hp*-finite element methods. Concretely, this means that for piecewise analytic functions, which appear, for example, as solutions of elliptic boundary and eigenvalue problems with analytic data, exponential approximation rates can be achieved. In other words, the number of parameters of neural networks to approximate such a function in the $W^{1,2}$-norm up to an error of ε is logarithmic in ε.

Theorem 3.2 requires the depth of the NN to grow. In fact, it can be shown that the same approximation rate cannot be achieved with shallow NNs. Indeed, there exists a certain optimal number of layers and, if the architecture has fewer layers than optimal, then the NNs need to have significantly more parameters, to achieve the same approximation fidelity. This has been observed in many different settings in [LS17, SS17, Yar17, PV18, EPGB19]. We state here the result of [Yar17]:

Theorem 3.3 (Depth-width approximation trade-off)**.** *Let $d, L \in \mathbb{N}$ with $L \geq 2$ and let $g \in C^2([0,1]^d)$ be a function which is not affine linear. Then there exists a constant $c \in (0,\infty)$ with the following property: For every $\varepsilon \in (0,1)$ and every ReLU NN architecture $a = (N, \varrho_R) = ((d, N_1, \dots, N_{L-1}, 1), \varrho_R)$ with L layers and $\|N\|_1 \leq c\varepsilon^{-1/(2(L-1))}$ neurons it holds that*

$$\inf_{\theta \in \mathbb{R}^{P(N)}} \|\Phi_a(\cdot,\theta) - g\|_{L^\infty([0,1]^d)} \geq \varepsilon.$$

This results is based on the observation that ReLU NNs are piecewise affine linear. The number of pieces they admit is linked to their capacity of approximating functions that have non-vanishing curvature. Using a construction similar to the example at the beginning of this subsection, it can be shown that the number of pieces that can be generated using an architecture $((1, N_1, \dots, N_{L-1}, 1), \varrho_R)$ scales roughly like $\prod_{\ell=1}^{L-1} N_\ell$.

In the framework of the aforementioned results, we can speak of a depth-width trade-off, see also Figure 3.3. A fine-grained estimate of achievable rates for freely varying depths has also been established in [She20].

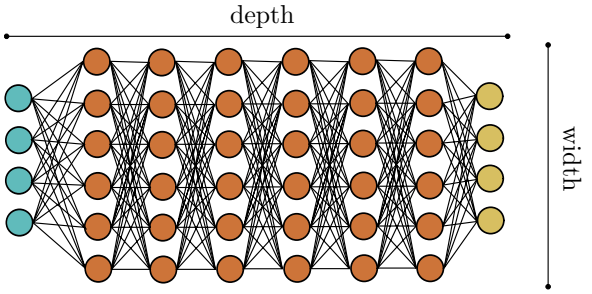


Figure 3.3: Standard feed-forward neural network. For certain approximation results, depth and width need to be in a fixed relationship to achieve optimal results.

3.3 Alternative notions of expressivity

Conceptual approaches to study the approximation power of deep NNs besides the classical approximation framework usually aim to relate structural properties of the NN to the "richness" of the set of possibly expressed functions. One early result in this direction is [MPCB14] which describes bounds on the number of *affine linear regions* of a ReLU NN $\Phi_{(N,\varrho_R)}(\cdot,\theta)$. In a simplified setting, we have seen estimates on the number of affine linear pieces already at the beginning of Subsection 3.2. Affine linear regions can be defined as the connected components of $\mathbb{R}^{N_0} \setminus H$, where H is the set of non-differentiability of the realization[20] $\Phi_{(N,\varrho_R)}(\cdot,\theta)$. A refined analysis on the number of such regions was, for example, conducted by [HvdG19]. It is found that deep ReLU neural networks can exhibit significantly more regions than their shallow counterparts.

[20] One can also study the potentially larger set of *activation regions* given by the connected components of $\mathbb{R}^{N_0} \setminus \big(\cup_{\ell=1}^{L-1} \cup_{i=1}^{N_\ell} H_{i,\ell} \big)$, where

$$H_{i,\ell} \coloneqq \{x \in \mathbb{R}^{N_0} : \Phi_i^{(\ell)}(x,\theta) = 0\},$$

with $\Phi_i^{(\ell)}$ as in (1.1), is the set of non-differentiability of the activation of the i-th neuron in the ℓ-th layer. In contrast to the linear regions, the activation regions are necessarily convex [RPK+17, HR19].

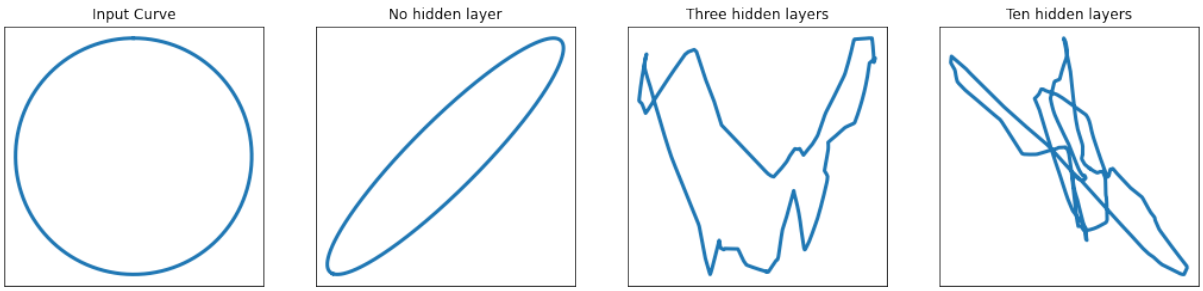


Figure 3.4: Shape of the trajectory $t \mapsto \Phi_{((2,n,\dots,n,2),\varrho_R)}(\gamma(t),\theta)$ of the output of a randomly initialized network with $0, 3, 10$ hidden layers. The input curve γ is the circle given in the leftmost image. The hidden layers have $n = 20$ neurons and the variance of the initialization is taken as $4/n$.

The reason for this effectiveness of depth is described by the following analogy: Through the ReLU each neuron $\mathbb{R}^d \ni x \mapsto \varrho_R(\langle x, w\rangle + b)$, $w \in \mathbb{R}^d$, $b \in \mathbb{R}$, splits the space into two affine linear regions separated by the hyperplane

$$\{x \in \mathbb{R}^d \colon \langle x, w\rangle + b = 0\}.$$

A shallow ReLU NN $\Phi_{((d,n,1),\varrho_R)}(\cdot,\theta)$ with n neurons in the hidden layer therefore produces a number of regions defined through n hyperplanes. Using classical bounds on the number of regions defined through hyperplane arrangements [Zas75], one can bound the number of affine linear regions by $\sum_{j=0}^{d}\binom{n}{j}$. Deepening neural networks then corresponds to a certain folding of the input space. Through this interpretation it can be seen that composing NNs can lead to a multiplication of the number of regions of the individual NNs resulting in an exponential efficiency of deep neural networks in generating affine linear regions[21].

This approach was further developed in [RPK+17] to a framework to study expressivity that to some extent allows to include the training phase. One central object studied in [RPK+17] are so-called *trajectory lengths*. In this context, one analyzes how the length of a non-constant curve in the input space changes in expectation through the layers of a NN. The authors find an exponential dependence of the expected curve length on the depth. Let us motivate this in the special case of a ReLU NN with architecture $a = ((N_0, n, \dots, n, N_L), \varrho_R)$ and depth $L \in \mathbb{N}$.

Given a non-constant continuous curve $\gamma \colon [0,1] \to \mathbb{R}^{N_0}$ in the input space, the length of the trajectory in the ℓ-th layer of the NN $\Phi_a(\cdot,\theta)$ is then given by

$$\operatorname{Length}(\bar{\Phi}^{(\ell)}(\gamma(\cdot),\theta)), \quad \ell \in [L-1],$$

where $\bar{\Phi}^{(\ell)}(\cdot,\theta)$ is the activation in the ℓ-th layer, see (1.1). Here the length of the curve is well-defined since $\bar{\Phi}^{(\ell)}(\cdot,\theta))$ is continuous and therefore $\bar{\Phi}^{(\ell)}(\gamma(\cdot),\theta)$ is continuous. Now, let the parameters Θ_1 of the NN Φ_a be initialized independently so that the entries corresponding to the weight matrices and bias vectors follow a normal distribution with zero mean and variances $1/n$ and 1, respectively. It is not hard to see, e.g., by Proposition 1.1, that the probability that $\bar{\Phi}^{(\ell)}(\cdot,\Theta_1)$ will map γ to a non-constant curve is positive and hence, for fixed $\ell \in [L-1]$,

$$\mathbb{E}\big[\operatorname{Length}(\bar{\Phi}^{(\ell)}(\gamma(\cdot),\Theta_1))\big] = c > 0.$$

Let $\sigma \in (0,\infty)$ and consider a second initialization Θ_σ, where we change the variances of the entries corresponding to the weight matrices and bias vectors to σ^2/n and σ^2, respectively. Recall that the ReLU is positively homogeneous, i.e., we have that $\varrho_R(\lambda x) = \lambda \varrho_R(x)$ for all $\lambda \in (0,\infty)$. Then it is clear that

$$\bar{\Phi}^{(\ell)}(\cdot,\Theta_\sigma) \sim \sigma^\ell \bar{\Phi}^{(\ell)}(\cdot,\Theta_1),$$

[21] However, to exploit this efficiency with respect to the depth, one requires highly oscillating pre-activations which in turn can only be achieved with a delicate selection of parameters. In fact, it can be shown that through random initialization the expected number of activation regions per unit cube depends mainly on the number of neurons in the NN, rather than its depth [HR19].

i.e., the activations corresponding to the two initialization strategies are identically distributed up to the factor σ^ℓ. Therefore, we immediately conclude that

$$\mathbb{E}\big[\operatorname{Length}(\bar{\Phi}^{(\ell)}(\gamma(\cdot), \Theta_\sigma))\big] = \sigma^\ell c.$$

This shows that the expected trajectory length depends exponentially on the depth of the NN, which is in line with the behavior of other notions of expressivity [PLR+16]. In [RPK+17] this result is also extended to the tanh activation function and the constant c is more carefully resolved. Empirically one also finds that the shapes of the trajectories become more complex in addition to becoming longer on average, see Figure 3.4.

4 Deep neural networks overcome the curse of dimensionality

In Subsection 1.3, one of the main puzzles of deep learning that we identified was the surprising performance of deep architectures on problems where the input dimensions are very high. This performance cannot be explained in the framework of classical approximation theory, since such results always suffer from the curse of dimensionality [Bel52, DeV98, NW09].

In this section, we present three approaches that offer explanations of this phenomenon. As before, we had to omit certain ideas which have been very influential in the literature to keep the length of this section under control. In particular, an important line of reasoning is that functions to be approximated often have compositional structures which NNs may approximate very well as reviewed in [PMR+17]. Note that also a suitable feature descriptor, factoring out invariances, might lead to a significantly reduced effective dimension, see Subsection 7.1.

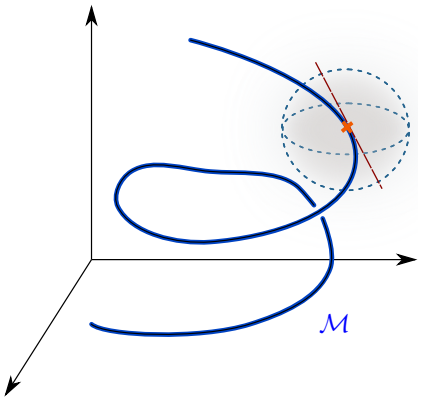


Figure 4.1: Illustration of a one-dimensional manifold $\mathcal{M}$ embedded in $\mathbb{R}^3$. For every point $x \in \mathcal{M}$ there exists a neighborhood in which the manifold can be linearly projected onto its tangent space at x such that the corresponding inverse function is differentiable.

4.1 Manifold assumption

A first remedy to the high-dimensional curse of dimensionality is what we call the *manifold assumption*. Here it is assumed that we are trying to approximate a function

$$g\colon \mathbb{R}^d \supset \mathcal{X} \to \mathbb{R},$$

where d is very large. However, we are not seeking to optimize with respect to the uniform norm or a regular L^p space, but instead consider a measure μ which is supported on a d'-dimensional manifold $\mathcal{M} \subset \mathcal{X}$. Then the error is measured in the $L^p(\mu)$-norm. Here we consider the case where $d' \ll d$. This setting is appropriate if the data $z = (x, y)$ of a prediction task is generated from a measure supported on $\mathcal{M} \times \mathbb{R}$.

This set-up or generalizations thereof have been fundamental in [CM18, SCC18, CJLZ19, SH19, CK20, NI20]. Let us describe an exemplary approach, where we consider locally C^k-regular functions and NNs with ReLU activation functions below:

1. *Describe the regularity of g on the manifold:* Naturally, we need to quantify the regularity of the function g restricted to $\mathcal{M}$ in an adequate way. The typical approach would be to make a definition via local coordinate charts. If we assume that $\mathcal{M}$ is an embedded submanifold of $\mathcal{X}$, then locally, i.e., in a neighborhood of a point $x \in \mathcal{M}$, the orthogonal projection of $\mathcal{M}$ onto the d'-dimensional tangent space $T_x\mathcal{M}$ is a diffeomorphism. The situation is depicted in Figure 4.1. Assuming $\mathcal{M}$ to be compact, we can choose a finite set of open balls $(U_i)_{i=1}^p$ that cover $\mathcal{M}$ and on which the local projections γ_i onto the respective tangent spaces as described above exists and are diffeomorphisms. Now we can define the regularity of g via classical regularity. In this example, we say that $g \in C^k(\mathcal{M})$ if $g \circ \gamma_i^{-1} \in C^k(\gamma_i(\mathcal{M} \cap U_i))$ for all $i \in [p]$.

2. *Construct localization and charts via neural networks:* According to the construction of local coordinate charts in Step 1, we can write g as follows:

$$g(x) = \sum_{i=1}^{p} \phi_i(x) \left(g \circ \gamma_i^{-1}(\gamma_i(x))\right) =: \sum_{i=1}^{p} \tilde{g}_i(\gamma_i(x), \phi_i(x)), \quad x \in \mathcal{M}, \tag{4.1}$$

where ϕ_i is a partition of unity such that $\operatorname{supp}(\phi_i) \subset U_i$. Note that γ_i is a linear map, hence representable by a one-layer NN. Since multiplication is a smooth operation, we have that if $g \in C^k(\mathcal{M})$ then $\tilde{g}_i \in C^k(\gamma_i(\mathcal{M} \cap U_i) \times [0,1])$.

The partition of unity ϕ_i needs to be emulated by NNs. For example, if the activation function is the ReLU, then such a partition can be efficiently constructed. Indeed, in [HLXZ20] it was shown that such NNs can represent linear finite elements exactly with fixed-size NNs and hence a partition of unity subordinate to any given covering of $\mathcal{M}$ can be constructed.

3. *Use a classical approximation result on the localized functions:* By some form of Whitney's extension theorem [Whi34], we can extend each $\tilde{g}_i$ to a function $\bar{g}_i \in C^k(\mathcal{X} \times [0,1])$ which by classical results can be approximated up to an error of $\varepsilon > 0$ by NNs of size $\mathcal{O}(\varepsilon^{-(d'+1)/k})$ for $\varepsilon \to 0$, see [Mha96, Yar17, SCC18].

4. *Use the compositionality of neural networks to build the final network:* We have seen that every component in the representation (4.1), i.e., $\tilde{g}_i$, γ_i, and ϕ_i can be efficiently represented by NNs. In addition, composition and summation are operations which can directly be implemented by NNs through increasing their depth and widening their layers. Hence (4.1) is efficiently—i.e., with a rate depending only on d' instead of the potentially much larger d—approximated by a NN.

Overall, we see that NNs are capable of learning local coordinate transformations and therefore reduce the complexity of a high-dimensional problem to the underlying low-dimensional problem given by the data distribution.

4.2 Random sampling

Already in 1992, Andrew Barron showed that under certain seemingly very natural assumptions on the function to approximate, a dimension-independent approximation rate by NNs can be achieved [Bar92, Bar93]. Specifically, the assumption is formulated as a condition on the Fourier transform of a function and the result is as follows.

Theorem 4.1 (Approximation of Barron-regular functions)**.** *Let $\varrho\colon \mathbb{R} \to \mathbb{R}$ be the ReLU or a sigmoidal function. Then there exists a constant $c \in (0,\infty)$ with the following property: For every $d, n \in \mathbb{N}$, every probability measure μ supported on $B_1(0) \subset \mathbb{R}^d$, and every $g \in L^1(\mathbb{R}^d)$ with $C_g := \int_{\mathbb{R}^d} \|\xi\|_2 |\hat{g}(\xi)| \,\mathrm{d}\xi < \infty$ it holds that*

$$\inf_{\theta \in \mathbb{R}^{P((d,n,1))}} \|\Phi_{((d,n,1),\varrho)}(\cdot,\theta) - g\|_{L^2(\mu)} \leq \frac{c}{\sqrt{n}} C_g,$$

Note that the L^2-approximation error can be replaced by an L^∞-estimate over the unit ball at the expense of a factor of the order of $\sqrt{d}$ on the right-hand side.

The key idea behind Theorem 4.1 is the following application of the law of large numbers: First, we observe that, per assumption, g can be represented via the inverse Fourier transform, as

$$\begin{aligned} g - g(0) &= \int_{\mathbb{R}^d} \hat{g}(\xi)(e^{2\pi i \langle \cdot, \xi \rangle} - 1) \,\mathrm{d}\xi \\ &= C_g \int_{\mathbb{R}^d} \frac{1}{\|\xi\|_2} (e^{2\pi i \langle \cdot, \xi \rangle} - 1) \frac{1}{C_g} \|\xi\|_2 \hat{g}(\xi) \,\mathrm{d}\xi \\ &= C_g \int_{\mathbb{R}^d} \frac{1}{\|\xi\|_2} (e^{2\pi i \langle \cdot, \xi \rangle} - 1) \,\mathrm{d}\mu_g(\xi), \end{aligned} \tag{4.2}$$

where μ_g is a probability measure. Then it is further shown in [Bar92] that there exist $(\mathbb{R}^d \times \mathbb{R})$-valued random variables $(\Xi, \widetilde{\Xi})$ such that (4.2) can be written as

$$g(x) - g(0) = C_g \int_{\mathbb{R}^d} \frac{1}{\|\xi\|_2} (e^{2\pi i \langle x, \xi \rangle} - 1) \, \mathrm{d}\mu_g(\xi) = C_g \mathbb{E}\big[\Gamma(\Xi, \widetilde{\Xi})(x)\big], \quad x \in \mathbb{R}^d, \tag{4.3}$$

where for every $\xi \in \mathbb{R}^d$, $\tilde{\xi} \in \mathbb{R}$ the function $\Gamma(\xi, \tilde{\xi}) \colon \mathbb{R}^d \to \mathbb{R}$ is given by

$$\Gamma(\xi, \tilde{\xi}) \coloneqq s(\xi, \tilde{\xi})(\mathbb{1}_{(0,\infty)}(-\langle \xi/\|\xi\|_2, \cdot \rangle - \tilde{\xi}) - \mathbb{1}_{(0,\infty)}(\langle \xi/\|\xi\|_2, \cdot \rangle - \tilde{\xi})) \quad \text{with} \quad s(\xi, \tilde{\xi}) \in \{-1, 1\}.$$

Now, let $((\Xi^{(i)}, \widetilde{\Xi}^{(i)}))_{i \in \mathbb{N}}$ be i.i.d. random variables with $(\Xi^{(1)}, \widetilde{\Xi}^{(1)}) \sim (\Xi, \widetilde{\Xi})$. Then, Bienaymé's identity and Fubini's theorem establish that

$$\begin{aligned} \mathbb{E}\left[\left\| g - g(0) - \frac{C_g}{n} \sum_{i=1}^{n} \Gamma(\Xi^{(i)}, \widetilde{\Xi}^{(i)}) \right\|_{L^2(\mu)}^2 \right] &= \int_{B_1(0)} \mathbb{V}\left[\frac{C_g}{n} \sum_{i=1}^{n} \Gamma(\Xi^{(i)}, \widetilde{\Xi}^{(i)})(x) \right] \mathrm{d}\mu(x) \\ &= \frac{C_g^2 \int_{B_1(0)} \mathbb{V}\big[\Gamma(\Xi, \widetilde{\Xi})(x)\big] \, \mathrm{d}\mu(x)}{n} \leq \frac{(2\pi C_g)^2}{n}, \end{aligned} \tag{4.4}$$

where the last inequality follows from combining (4.3) with the fact that $|e^{2\pi i \langle x, \xi \rangle} - 1| / \|\xi\|_2 \leq 2\pi$, $x \in B_1(0)$.

This implies that there exists a realization $((\xi^{(i)}, \tilde{\xi}^{(i)}))_{i \in \mathbb{N}}$ of the random variables $((\Xi^{(i)}, \widetilde{\Xi}^{(i)}))_{i \in \mathbb{N}}$ that achieves L^2-approximation error of $n^{-1/2}$. Therefore, it remains to show that NNs can well approximate the functions $((\Gamma(\xi^{(i)}, \tilde{\xi}^{(i)}))_{i \in \mathbb{N}}$. Now it is not hard to see that the function $\mathbb{1}_{(0,\infty)}$ and hence functions of the form $\Gamma(\xi, \tilde{\xi})$, $\xi \in \mathbb{R}^d$, $\tilde{\xi} \in \mathbb{R}$, can be arbitrarily well approximated with a fixed-size, two-layer NN with a sigmoidal or ReLU activation function. Thus, we obtain an approximation rate of $n^{-1/2}$ when approximating functions with one finite Fourier moment by two-layer NNs with n hidden neurons.

It was pointed out already in the dissertation of Emmanuel Candès [Can98] that the approximation rate of NNs for Barron-regular functions is also achievable by n-term approximation with complex exponentials, as is apparent by considering (4.2). However, for deeper NNs, the results also extend to high-dimensional non-smooth functions, where Fourier-based methods are certain to suffer from the curse of dimensionality [CPV20].

In addition, the random sampling idea above was extended in [EMW19b, EMWW20, EW20b, EW20c] to facilitate dimension-independent approximation of vastly more general function spaces. Basically, the idea is to use (4.3) as an inspiration and define the *generalized Barron space* as all functions that may be represented as

$$\mathbb{E}\big[\mathbb{1}_{(0,\infty)}(\langle \Xi, \cdot \rangle - \widetilde{\Xi})\big]$$

for any random variable $(\Xi, \widetilde{\Xi})$. In this context, deep and compositional versions of Barron spaces were introduced and studied in [BK18, EMW19a, EW20a], which considerably extend the original theory.

4.3 PDE assumption

Another structural assumption that leads to the absence of the curse of dimensionality in some cases is that the function we are trying to approximate is given as the solution to a partial differential equation. It is by no means clear that this assumption leads to approximation without the curse of dimensionality, since most standard methods, such as finite elements, sparse grids, or spectral methods typically suffer from the curse of dimensionality.

This is not merely an abstract theoretical problem: Very recently, in [AHNB$^+$20] it was shown that two different gold standard methods for solving the multi-electron Schrödinger equation produce completely different interaction energy predictions when applied to large delocalized molecules. Classical numerical representations are simply not expressive enough to accurately represent complicated high-dimensional structures such as wave functions with long-range interactions.

Interestingly, there exists an emerging body of work that shows that NNs do not suffer from these shortcomings and enjoy superior expressivity properties as compared to standard numerical representations.

Such results include, for example, [GHJVW20, GS20, HJKN20] for (linear and semilinear) parabolic evolution equations, [GH22] for stationary elliptic PDEs, [GH21] for nonlinear Hamilton–Jacobi–Bellman equations, or [KPRS19] for parametric PDEs. In all these cases, the absence of the curse of dimensionality in terms of the theoretical approximation power of NNs could be rigorously established.

One way to prove such results is via stochastic representations of the PDE solutions, as well as associated sampling methods. We illustrate the idea for the simple case of linear Kolmogorov PDEs, that is the problem of representing the function $g\colon \mathbb{R}^d \times [0,\infty) \to \mathbb{R}$ satisfying[22]

$$\frac{\partial g}{\partial t}(x,t) = \frac{1}{2}\mathrm{Tr}\big(\sigma(x,t)[\sigma(x,t)]^* \nabla_x^2 g(x,t)\big) + \langle \mu(x,t), \nabla_x g(x,t)\rangle, \quad g(x,0) = \varphi(x), \tag{4.5}$$

where the functions

$$\varphi\colon \mathbb{R}^d \to \mathbb{R} \quad \text{(initial condition)} \quad \text{and} \quad \sigma\colon \mathbb{R}^d \to \mathbb{R}^{d\times d}, \quad \mu\colon \mathbb{R}^d \to \mathbb{R}^d \quad \text{(coefficient functions)}$$

are continuous and satisfy suitable growth conditions. A stochastic representation of g is given via the Ito processes $(\mathcal{S}_{x,t})_{t\geq 0}$ satisfying

$$d\mathcal{S}_{x,t} = \mu(\mathcal{S}_{x,t})dt + \sigma(\mathcal{S}_{x,t})dB_t, \quad \mathcal{S}_{x,0} = x, \tag{4.6}$$

where $(B_t)_{t\geq 0}$ is a d-dimensional Brownian motion. Then g is described via the Feynman–Kac formula which states that

$$g(x,t) = \mathbb{E}[\varphi(\mathcal{S}_{x,t})], \quad x \in \mathbb{R}^d,\ t \in [0,\infty). \tag{4.7}$$

Roughly speaking, a NN approximation result can be proven by first approximating, via the law of large numbers,

$$g(x,t) = \mathbb{E}[\varphi(\mathcal{S}_{x,t})] \approx \frac{1}{n}\sum_{i=1}^{n} \varphi(\mathcal{S}_{x,t}^{(i)}), \tag{4.8}$$

where $(\mathcal{S}_{x,t}^{(i)})_{i=1}^n$ are i.i.d. random variables with $\mathcal{S}_{x,t}^{(1)} \sim \mathcal{S}_{x,t}$. Care has to be taken to establish such an approximation *uniformly in the computational domain*, for example, for every (x,t) in the unit cube $[0,1]^d \times [0,1]$, see (4.4) for a similar estimate and [GHJVW20, GS20] for two general approaches to ensure this property. Aside from this issue, (4.8) represents a standard Monte Carlo estimator which can be shown to be free of the curse of dimensionality.

As a next step, one needs to establish that realizations of the processes $(x,t) \mapsto \mathcal{S}_{x,t}$ can be efficiently approximated by NNs. This can be achieved by emulating a suitable time-stepping scheme for the SDE (4.6) by NNs which, roughly speaking, can be done without incurring the curse of dimensionality whenever the coefficient functions μ, σ can be approximated by NNs without incurring the curse of dimensionality and some growth conditions hold true. In a last step one assumes that the initial condition φ can be approximated by NNs without incurring the curse of dimensionality which, by the compositionality of NNs and the previous step, directly implies that realizations of the processes $(x,t) \mapsto \varphi(\mathcal{S}_{x,t})$ can be approximated by NNs without incurring the curse of dimensionality. By (4.8) this implies a corresponding approximation result for the solution of the Kolmogorov PDE g in (4.5).

Informally, we have discovered a regularity result for linear Kolmogorov equations, namely that (modulo some technical conditions on μ, σ), *the solution g of* (4.5) *can be approximated by NNs without incurring the curse of dimensionality whenever the same holds true for the initial condition φ, as well as the coefficient functions μ, σ*. In other words, *the property of being approximable by NNs without curse of dimensionality is preserved under the flow induced by the PDE* (4.5). Some comments are in order:

[22] The natural solution concept to this type of PDEs is the viscosity solution concept, a thorough study of which can be found in [HHJ15].

Assumption on the initial condition: One may wonder if the assumption that the initial condition φ can be approximated by NNs without incurring the curse of dimensionality is justified. This is at least the case in many applications in computational finance where the function φ typically represents an option pricing formula and (4.5) represents the famous Black–Scholes model. It turns out that nearly all common option pricing formulas are constructed from iterative applications of linear maps and maximum/minimum functions—in other words, in many applications in computational finance, the initial condition φ can be *exactly* represented by a small ReLU NN.

Generalization and optimization error: The Feynman–Kac representation (4.7) directly implies that $g(\cdot, t)$ can be computed as the Bayes optimal function of a regression task with input features $X \sim \mathcal{U}([0,1]^d)$ and labels $Y = \varphi(\mathcal{S}_{X,t})$, which allows for an analysis of the generalization error as well as implementations based on ERM algorithms [BGJ20, BBG+21].

While it is in principle possible to analyze the approximation and generalization error, the analysis of the computational cost and/or convergence of corresponding SGD algorithms is completely open. Some promising numerical results exist, see, for instance, Figure 4.2, but the stable training of NNs approximating PDEs to very high accuracy (that is needed in several applications such as quantum chemistry) remains very challenging. The recent work [GV21] has even proven several impossibility results in that direction.

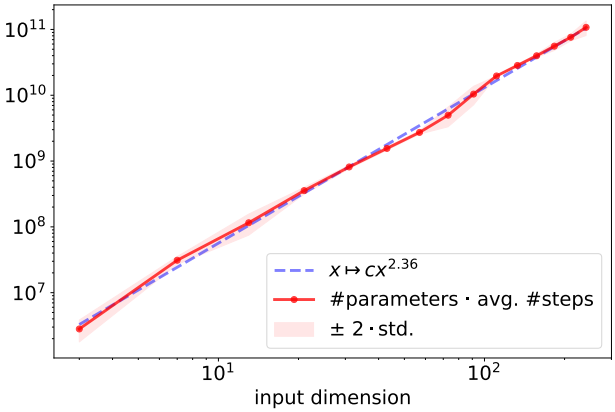


Figure 4.2: Computational complexity as number of neural network parameters times number of SGD steps to solve heat equations of varying dimensions up to a specified precision. According to the fit above, the scaling is polynomial in the dimension [BDG20].

Extensions and abstract idea: Similar techniques may be used to prove expressivity results for nonlinear PDEs, for example, using nonlinear Feynman–Kac-type representations of [PP92] in place of (4.7) and multilevel Picard sampling algorithms of [EHJK19] in place of (4.8).

We can also formulate the underlying idea in an abstract setting (a version of which has also been used in Subsection 4.2). Assume that a high-dimensional function $g\colon \mathbb{R}^d \to \mathbb{R}$ admits a probabilistic representation of the form

$$g(x) = \mathbb{E}[Y_x], \quad x \in \mathbb{R}^d, \tag{4.9}$$

for some random variable Y_x which can be approximated by an iterative scheme

$$\mathcal{Y}_x^{(L)} \approx Y_x \quad \text{and} \quad \mathcal{Y}_x^{(\ell)} = T_\ell(\mathcal{Y}_x^{(\ell-1)}), \quad \ell = 1, \dots, L,$$

with dimension-independent convergence rate. If we can approximate realizations of the initial mapping $x \mapsto \mathcal{Y}_x^0$ and the maps T_ℓ, $\ell \in [L]$, by NNs and the numerical scheme is stable enough, then we can also approximate $\mathcal{Y}_x^{(L)}$ using compositionality. Emulating a uniform Monte-Carlo approximator of (4.9) then leads to approximation results for g without curse of dimensionality. In addition, one can choose a $\mathbb{R}^d$-valued random variable X as input features and define the corresponding labels by Y_X to obtain a prediction task, which can be solved by means of ERM.

Other methods: There exist a number of additional works related to the approximation capacities of NNs for high-dimensional PDEs, for example, [EGJS18, LTY19, SZ19]. In most of these works, the proof technique consists of emulating an existing method that does not suffer from the curse of dimensionality. For instance, in the case of first-order transport equations, one can show in some cases that NNs are capable of emulating the method of characteristics, which then also yields approximation results that are free of the curse of dimensionality [LP21].

5 Optimization of deep neural networks

We recall from Subsections 1.3 and 1.2.1 that the standard algorithm to solve the empirical risk minimization problem over the hypothesis set of NNs is stochastic gradient descent. This method would be guaranteed to converge to a global minimum of the objective if the empirical risk were convex, viewed as a function of the NN parameters. However, this function is severely nonconvex, may exhibit (higher-order) saddle points, seriously suboptimal local minima, and wide flat areas where the gradient is very small.

On the other hand, in applications, excellent performance of SGD is observed. This indicates that the trajectory of the optimization routine somehow misses suboptimal critical points and other areas that may lead to slow convergence. Clearly, the classical theory does not explain this performance. Below we describe some exemplary novel approaches that give partial explanations of this success.

In the flavor of this article, the aim of this section is to present some selected ideas rather than giving an overview of the literature. To give at least some detail about the underlying ideas and to keep the length of this section reasonable, a selection of results had to be made and some ground-breaking results had to be omitted.

5.1 Loss landscape analysis

Given a NN $\Phi(\cdot, \theta)$ and training data $s \in \mathcal{Z}^m$ the function $\theta \mapsto r(\theta) \coloneqq \widehat{\mathcal{R}}_s(\Phi(\cdot, \theta))$ describes, in a natural way, through its graph, a high-dimensional surface. This surface may have regions associated with lower values of $\widehat{\mathcal{R}}_s$ which resemble valleys of a landscape if they are surrounded by regions of higher values. The analysis of the topography of this surface is called *loss landscape analysis*. Below we shall discuss a couple of approaches that yield deep insights into the shape of this landscape.

Spin glass interpretation: One of the first discoveries about the shape of the loss landscape comes from deep results in statistical physics. The Hamiltonian of the *spin glass model* is a random function on the $(n-1)$-dimensional sphere of radius $\sqrt{n}$. Making certain simplifying assumptions, it was shown in [CHM+15] that the loss of a NN with random inputs can be considered as the Hamiltonian of a spin glass model, where the inputs of the model are the parameters of the NN.

This connection has far-reaching implications for the loss landscape of NNs because of the following surprising property of the Hamiltonian of spin glass models: Consider the set of critical points of this set, and associate to each point an *index* that denotes the percentage of the eigenvalues of the Hessian at that point which are negative. This index corresponds to the relative number of directions in which the loss landscape has negative curvature. Then with high probability, a picture like we see in Figure 5.1 emerges [AAČ13]. More precisely, the further away from the optimal loss we are, the more unstable the critical points become. Conversely, if one finds oneself in a local minimum, it is reasonable to assume that the loss is close to the global minimum.

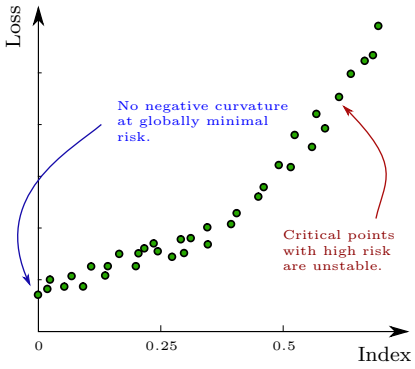


Figure 5.1: Sketch of the distribution of critical points of the Hamiltonian of a spin glass model.

While some of the assumptions establishing the connection between the spin glass model and NNs are unrealistic in practice [CLA15], the theoretical distribution of critical points as in Figure 5.1 is visible in many practical applications [DPG+14].

Paths and level sets: Another line of research is to understand the loss landscape by analyzing paths through the parameter space. In particular, the existence of paths in parameter space, such that the associated empirical risks are monotone along the path. Surely, should there exist a path of nonincreasing empirical risk from every point to the global minimum, then we can be certain that no non-global minima exist, since no

such path can escape a minimum. An even stronger result holds. In fact, the existence of such paths shows that the loss landscape has connected level sets [FB17, VBB19].

A crucial ingredient of the analysis of such paths are *linear substructures*. Consider a biasless two-layer NN Φ of the form

$$\mathbb{R}^d \ni x \mapsto \Phi(x,\theta) \coloneqq \sum_{j=1}^{n} \theta_j^{(2)} \varrho\big(\langle \theta_j^{(1)}, \begin{bmatrix} x \\ 1 \end{bmatrix} \rangle\big), \tag{5.1}$$

where $\theta_j^{(1)} \in \mathbb{R}^{d+1}$ for $j \in [n]$, $\theta^{(2)} \in \mathbb{R}^n$, ϱ is a Lipschitz continuous activation function, and we augment the vector x by a constant 1 in the last coordinate as outlined in Remark 1.5. If we consider $\theta^{(1)}$ to be fixed, then it is clear that the space

$$\widetilde{\mathcal{F}}_{\theta^{(1)}} \coloneqq \{\Phi(\cdot,\theta)\colon \theta = (\theta^{(1)},\theta^{(2)}),\ \theta^{(2)} \in \mathbb{R}^n\}$$

is a linear space. If the risk[23] is convex, as is the case for the widely used quadratic or logistic loss, then this implies that $\theta^{(2)} \mapsto r\big((\theta^{(1)},\theta^{(2)})\big)$ is a convex map and hence, for every parameter set $\mathcal{P} \subset \mathbb{R}^n$ this map assumes its maximum on $\partial\mathcal{P}$. Therefore, within the vast parameter space, there are many paths traveling along which does not increase the risk above the risk of the start and end points.

This idea was, for example, used in [FB17] in a way similar to the following simple sketch: Assume that, for two parameters θ and $\theta_{\min}$ there exists a linear subspace of NNs $\widetilde{\mathcal{F}}_{\hat\theta^{(1)}}$ such that there are paths γ_1 and γ_2 connecting $\Phi(\cdot,\theta)$ and $\Phi(\cdot,\theta_{\min})$ to $\widetilde{\mathcal{F}}_{\hat\theta^{(1)}}$ respectively. Further assume that the paths are such that along γ_1 and γ_2 the risk does not significantly exceed $\max\{r(\theta), r(\theta_{\min})\}$. Figure 5.2 shows a visualization of these paths. In this case, a path from θ to $\theta_{\min}$ not significantly exceeding $r(\theta)$ along the way is found by concatenating the paths γ_1, a path along $\widetilde{\mathcal{F}}_{\hat\theta^{(1)}}$, and γ_2. By the previous discussion, we know that only γ_1 and γ_2 determine the extent to which the combined path exceeds $r(\theta)$ along its way. Hence, we need to ask about the existence of $\widetilde{\mathcal{F}}_{\hat\theta^{(1)}}$ that facilitates the construction of appropriate γ_1 and γ_2.

To understand why a good choice of $\widetilde{\mathcal{F}}_{\hat\theta^{(1)}}$, so that the risk along γ_1 and γ_2 will not rise much higher than $r(\theta)$, is likely possible, we set[24]

$$\hat\theta_j^{(1)} \coloneqq \begin{cases} \theta_j^{(1)} & \text{for } j \in [n/2], \\ (\theta_{\min}^{(1)})_j & \text{for } j \in [n] \setminus [n/2]. \end{cases}$$

In other words, the first half of $\hat\theta^{(1)}$ is made from $\theta^{(1)}$ and the second from $\theta_{\min}^{(1)}$. If $\theta_j^{(1)}$, $j \in [N]$, are realizations of random variables distributed uniformly on the d-dimensional unit sphere, then by invoking standard covering bounds of spheres (e.g., [Ver18, Corollary 4.2.13]), we expect that, for $\varepsilon > 0$ and a sufficiently large number of neurons n, the vectors $(\theta_j^{(1)})_{j=1}^{n/2}$ already ε-approximate all vectors $(\theta_j^{(1)})_{j=1}^{n}$. Replacing all vectors $(\theta_j^{(1)})_{j=1}^{n}$ by their nearest neighbor in $(\theta_j^{(1)})_{j=1}^{n/2}$ can be done with a linear path in the parameter space, and, given that r is locally Lipschitz continuous and $\|\theta^{(2)}\|_1$ is bounded, this operation will not increase the risk by more than $\mathcal{O}(\varepsilon)$. We denote the vector resulting from this replacement procedure by $\theta_*^{(1)}$. Since for all $j \in [n] \setminus [n/2]$ we now have that

$$\varrho\big(\langle (\theta_*^{(1)})_j, \begin{bmatrix} \cdot \\ 1 \end{bmatrix} \rangle\big) \in \left\{ \varrho\big(\langle (\theta_*^{(1)})_k, \begin{bmatrix} \cdot \\ 1 \end{bmatrix} \rangle\big)\colon k \in [n/2] \right\},$$

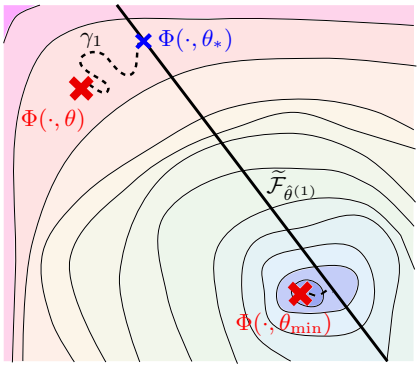


Figure 5.2: Construction of a path from an initial point θ to the global minimum $\theta_{\min}$ that does not have significantly higher risk than the initial point along the way. We depict here the landscape as a function of the neural network realizations instead of their parametrizations so that this landscape is convex.

[23] As most statements in this subsection are valid for the empirical risk $r(\theta) = \widehat{\mathcal{R}}_s(\Phi(\cdot,\theta))$ as well as the risk $r(\theta) = \mathcal{R}(\Phi(\cdot,\theta))$, given a suitable distribution of Z, we will just call r the risk.

[24] We assume w.l.o.g. that n is a multiple of 2.

there exists a vector $\theta_*^{(2)}$ with $(\theta_*^{(2)})_j = 0$, $j \in [n] \setminus [n/2]$, so that

$$\Phi(\cdot, (\theta_*^{(1)}, \theta^{(2)})) = \Phi(\cdot, (\theta_*^{(1)}, \lambda\theta_*^{(2)} + (1-\lambda)\theta^{(2)})), \quad \lambda \in [0,1].$$

In particular, this path does not change the risk between $(\theta_*^{(1)}, \theta^{(2)})$ and $(\theta_*^{(1)}, \theta_*^{(2)})$. Now, since $(\theta_*^{(2)})_j = 0$ for $j \in [n] \setminus [n/2]$, the realization $\Phi(\cdot, (\theta_*^{(1)}, \theta_*^{(2)}))$ is computed by a sub-network consisting of the first $n/2$ hidden neurons and we can replace the parameters corresponding to the other neurons without any effect on the realization function. Specifically, it holds that

$$\Phi(\cdot, (\theta_*^{(1)}, \theta_*^{(2)})) = \Phi(\cdot, (\lambda\hat{\theta}^{(1)} + (1-\lambda)\theta_*^{(1)}, \theta_*^{(2)})), \quad \lambda \in [0,1],$$

yielding a path of constant risk between $(\theta_*^{(1)}, \theta_*^{(2)})$ and $(\hat{\theta}^{(1)}, \theta_*^{(2)})$. Connecting these paths completes the construction of γ_1 and shows that the risk along γ_1 does not exceed that at θ by more than $\mathcal{O}(\varepsilon)$. Of course, γ_2 can be constructed in the same way. The entire construction is depicted in Figure 5.2.

Overall, this derivation shows that for sufficiently wide NNs (appropriately randomly initialized) it is very likely possible to almost connect a random parameter value to the global minimum with a path which along the way does not need to climb much higher than the initial risk.

In [VBB19], a similar approach is taken and the convexity in the last layer is used. However, the authors invoke the concept of intrinsic dimension to elegantly solve the non-linearity of $r((\theta^{(1)}, \theta^{(2)}))$ with respect to $\theta^{(1)}$. Additionally, [SS16] constructs a path of decreasing risk from random initializations. The idea here is that if one starts at a point of sufficiently high risk, one can always find a path to the global optimum with strictly decreasing risk. The intriguing insight behind this result is that if the initialization is sufficiently bad, i.e., worse than that of a NN outputting only zero, then there exist two operations that influence the risk directly. Multiplying the last layer with a number smaller than one will decrease the risk, whereas the opposite will increase it. Using this tuning mechanism, any given potentially non-monotone path from the initialization to the global minimum can be modified so that it is strictly monotonically decreasing. In a similar spirit, [NH17] shows that if a deep NN has a layer with more neurons than training data points, then under certain assumptions the training data will typically be mapped to linearly independent points in that layer. Of course, this layer could then be composed with a linear map that maps the linearly independent points to any desirable output, in particular one that achieves vanishing empirical risk, see also Proposition 1.1. As for two-layer NNs, the previous discussion on linear paths immediately shows that in this situation a monotone path to the global minimum exists.

5.2 Lazy training and provable convergence of stochastic gradient descent

When training highly overparametrized NNs, one often observes that the parameters of the NNs barely change during training. In Figure 5.3, we show the relative distance that the parameters travel through the parameter space during the training of NNs of varying numbers of neurons per layer.

The effect described above has been observed repeatedly and theoretically explained, see, e.g., [DZPS18, LL18, AZLS19, DLL$^+$19, ZCZG20]. In Subsection 2.1, we have already seen a high-level overview and, in particular, the function space perspective of this phenomenon in the infinite width limit. Below we present a short and highly simplified derivation of this effect and show how it leads to provable convergence of gradient descent for sufficiently overparametrized deep NNs.

A simple learning model: We consider again the simple NN model of (5.1) with a smooth activation function ϱ which is not affine linear. For the quadratic loss and training data $s = ((x^{(i)}, y^{(i)}))_{i=1}^m \in (\mathbb{R}^d \times \mathbb{R})^m$, where $x_i \neq x_j$ for all $i \neq j$, the empirical risk is given by

$$r(\theta) = \widehat{\mathcal{R}}_s(\theta) = \frac{1}{m}\sum_{i=1}^m (\Phi(x^{(i)}, \theta) - y^{(i)})^2.$$

Let us further assume that $\Theta_j^{(1)} \sim \mathcal{N}(0, 1/n)^{d+1}$, $j \in [n]$, and $\Theta_j^{(2)} \sim \mathcal{N}(0, 1/n)$, $j \in [n]$, are independent random variables.

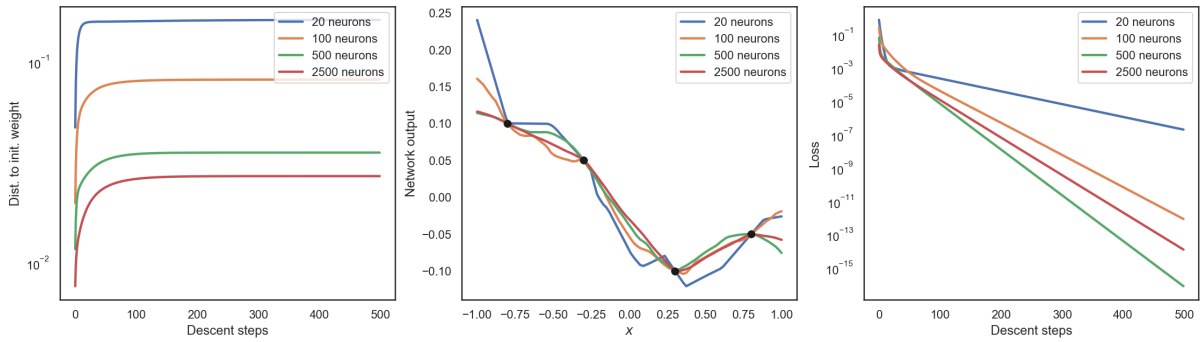


Figure 5.3: Four networks with architecture $((1, n, n, 1), \varrho_R)$ and $n \in \{20, 100, 500, 2500\}$ neurons per hidden layer were trained by gradient descent to fit four points that are shown in the middle figure as black dots. We depict on the left the relative Euclidean distance of the parameters from the initialization through the training process. In the middle, we show the final trained NNs. On the right we show the behavior of the training error.

A peculiar kernel: Next, we would like to understand how the gradient $\nabla_\theta r(\Theta)$ looks like with high probability over the initialization $\Theta = (\Theta^{(1)}, \Theta^{(2)})$. Similar to (2.3), we have by restricting the gradient to $\theta^{(2)}$ and applying the chain rule that

$$\begin{aligned} \|\nabla_\theta r(\Theta)\|_2^2 &\geq \frac{4}{m^2} \Big\| \sum_{i=1}^m \nabla_{\theta^{(2)}} \Phi(x^{(i)}, \Theta)(\Phi(x^{(i)}, \Theta) - y^{(i)}) \Big\|_2^2 \\ &= \frac{4}{m^2} \big((\Phi(x^{(i)}, \Theta) - y^{(i)})_{i=1}^m\big)^T \bar{K}_\Theta (\Phi(x^{(j)}, \Theta) - y^{(j)})_{j=1}^m, \end{aligned} \tag{5.2}$$

where $\bar{K}_\Theta$ is a random $\mathbb{R}^{m \times m}$-valued kernel given by

$$(\bar{K}_\Theta)_{i,j} := \big(\nabla_{\theta^{(2)}} \Phi(x^{(i)}, \Theta)\big)^T \nabla_{\theta^{(2)}} \Phi(x^{(j)}, \Theta), \quad i, j \in [m].$$

This kernel is closely related to the neural tangent kernel in (2.4) evaluated at the features $(x^{(i)})_{i=1}^m$ and the random initialization Θ. It is a slightly simplified version thereof, as in (2.4) the gradient is taken with respect to the full vector θ. This can also be regarded as the kernel associated with a *random features model* [RR+07].

Note that for our two-layer NN we have that

$$\big(\nabla_{\theta^{(2)}} \Phi(x, \Theta)\big)_k = \varrho\left(\left\langle \Theta_k^{(1)}, \begin{bmatrix} x \\ 1 \end{bmatrix} \right\rangle\right), \quad x \in \mathbb{R}^d, \ k \in [n].$$

Thus, we can write $\bar{K}_\Theta$ as the following sum of (random) rank one matrices:

$$\bar{K}_\Theta = \sum_{k=1}^n v_k v_k^T \qquad \text{with} \quad v_k = \left(\varrho\left(\left\langle \Theta_k^{(1)}, \begin{bmatrix} x^{(i)} \\ 1 \end{bmatrix} \right\rangle\right)\right)_{i=1}^m \in \mathbb{R}^m, \quad k \in [n]. \tag{5.3}$$

The kernel $\bar{K}_\Theta$ are symmetric and positive semi-definite by construction. It is positive definite if it is non-singular, i.e., if at least m of the n vectors v_k, $k \in [n]$, are linearly independent. Proposition 1.1 shows that for $n = m$ the probability of that event is not zero, say δ, and is therefore at least $1 - (1 - \delta)^{\lfloor n/m \rfloor}$ for arbitrary n. In other words, the probability increases rapidly with n. It is also clear from (5.3) that $\mathbb{E}[\bar{K}_\Theta]$ scales linearly with n.

From this intuitive derivation, we conclude that for sufficiently large n, with high probability $\bar{K}_\Theta$ is a positive definite kernel with smallest eigenvalue $\lambda_{\min}(\bar{K}_\Theta)$ scaling linearly with n. The properties of $\bar{K}_\Theta$, in particular its positive definiteness, have been studied much more rigorously as already described in Subsection 2.1.

Control of the gradient: Applying the expected behavior of the smallest eigenvalue $\lambda_{\min}(\bar{K}_\Theta)$ of $\bar{K}_\Theta$ to (5.2), we conclude that with high probability

$$\|\nabla_\theta r(\Theta)\|_2^2 \geq \frac{4}{m^2}\lambda_{\min}(\bar{K}_\Theta)\|(\Phi(x^{(i)},\Theta)-y^{(i)})_{i=1}^m\|_2^2 \gtrsim \frac{n}{m}r(\Theta). \tag{5.4}$$

To understand what will happen when applying gradient descent, we first need to understand how the situation changes in a neighborhood of Θ. We fix $x \in \mathbb{R}^d$ and observe that by the mean value theorem for all $\bar{\theta} \in B_1(0)$ we have

$$\big\|\nabla_\theta \Phi(x,\Theta) - \nabla_\theta \Phi(x,\Theta+\bar{\theta})\big\|_2^2 \lesssim \sup_{\hat{\theta}\in B_1(0)} \big\|\nabla_\theta^2 \Phi(x,\Theta+\hat{\theta})\big\|_{\mathrm{op}}^2, \tag{5.5}$$

where $\|\nabla_\theta^2\Phi(x,\Theta+\hat{\theta})\|_{\mathrm{op}}$ denotes the operator norm of the Hessian of $\Phi(x,\cdot)$ at $\Theta+\hat{\theta}$. From inspection of (5.1), it is not hard to see that for all $i,j\in[n]$ and $k,\ell\in[d+1]$

$$\mathbb{E}\left[\Big(\frac{\partial^2\Phi(x,\Theta)}{\partial\theta_i^{(2)}\partial\theta_j^{(2)}}\Big)^2\right]=0, \quad \mathbb{E}\left[\Big(\frac{\partial^2\Phi(x,\Theta)}{\partial\theta_i^{(2)}\partial(\theta_j^{(1)})_k}\Big)^2\right]\lesssim \delta_{i,j}, \quad \text{and} \quad \mathbb{E}\left[\Big(\frac{\partial^2\Phi(x,\Theta)}{\partial(\theta_i^{(1)})_k\partial(\theta_j^{(1)})_\ell}\Big)^2\right]\lesssim \frac{\delta_{i,j}}{n},$$

where $\delta_{i,j}=0$ if $i\neq j$ and $\delta_{i,i}=1$ for all $i,j\in[n]$. For sufficiently large n, we have that $\nabla_\theta^2\Phi(x,\Theta)$ is in expectation approximately a block band matrix with band-width $d+1$. Therefore, we conclude that $\mathbb{E}\big[\|\nabla_\theta^2\Phi(x,\Theta)\|_{\mathrm{op}}^2\big]\lesssim 1$. Hence, we obtain by concentration of Gaussian random variables that with high probability $\|\nabla_\theta^2\Phi(x,\Theta)\|_{\mathrm{op}}^2\lesssim 1$. By the block-banded form of $\nabla_\theta^2\Phi(x,\Theta)$ we have that, even after perturbation of Θ by a vector $\hat{\theta}$ with norm bounded by 1, the term $\|\nabla_\theta^2\Phi(x,\Theta+\hat{\theta})\|_{\mathrm{op}}^2$ is bounded, which yields that the right-hand side of (5.5) is bounded with high probability.

Using (5.5), we can extend (5.4), which holds with high probability, to a neighborhood of Θ by the following argument: Let $\bar{\theta}\in B_1(0)$, then

$$\begin{aligned}
\|\nabla_\theta r(\Theta+\bar{\theta})\|_2^2 &\geq \frac{4}{m^2}\Big\|\sum_{i=1}^m \nabla_{\theta^{(2)}}\Phi(x^{(i)},\Theta+\bar{\theta})(\Phi(x^{(i)},\Theta+\bar{\theta})-y^{(i)})\Big\|_2^2\\
&\underset{(5.5)}{=} \frac{4}{m^2}\Big\|\sum_{i=1}^m (\nabla_{\theta^{(2)}}\Phi(x^{(i)},\Theta)+\mathcal{O}(1))(\Phi(x^{(i)},\Theta+\bar{\theta})-y^{(i)})\Big\|_2^2\\
&\underset{(*)}{\gtrsim} \frac{1}{m^2}(\lambda_{\min}(\bar{K}_\Theta)+\mathcal{O}(1))\|(\Phi(x^{(i)},\Theta+\bar{\theta})-y^{(i)})_{i=1}^m\|_2^2\\
&\gtrsim \frac{n}{m}r(\Theta+\bar{\theta}),
\end{aligned} \tag{5.6}$$

where the estimate marked by $(*)$ uses the positive definiteness of $\bar{K}_\Theta$ again and only holds for sufficiently large n, so that the $\mathcal{O}(1)$ term is negligible.

We conclude that, with high probability over the initialization Θ, on a ball of fixed radius around Θ the squared Euclidean norm of the gradient of the empirical risk is lower bounded by $\frac{n}{m}$ times the empirical risk.

Exponential convergence of gradient descent: For sufficiently small step sizes η, the observation in the previous paragraph yields the following convergence rate for gradient descent as in Algorithm 1, specifically (1.5), with $m'=m$ and $\Theta^{(0)}=\Theta$: If $\|\Theta^{(k)}-\Theta\|\leq 1$ for all $k\in[K+1]$, then[25]

$$r(\Theta^{(K+1)}) \approx r(\Theta^{(K)}) - \eta\|\nabla_\theta r(\Theta^{(K)})\|_2^2 \leq \Big(1-\frac{c\eta n}{m}\Big)r(\Theta^{(K)}) \lesssim \Big(1-\frac{c\eta n}{m}\Big)^K, \tag{5.7}$$

for $c\in(0,\infty)$ so that $\|\nabla_\theta r(\Theta^{(k)})\|_2^2\geq \frac{cn}{m}r(\Theta^{(k)})$ for all $k\in[K]$.

[25]Note that the step-size η needs to be small enough to facilitate the approximation step in (5.7). Hence, we cannot simply put $\eta=m/(cn)$ in (5.7) and have convergence after one step.

Let us assume without proof that the estimate (5.6) could be extended to an equivalence. In other words, we assume that we additionally have that $\|\nabla_\theta r(\Theta+\bar{\theta})\|_2^2 \lesssim \frac{n}{m} r(\Theta+\bar{\theta})$. This, of course, could be shown with similar tools as were used for the lower bound. Then we have that $\|\Theta^{(k)}-\Theta\|_2 \leq 1$ for all $k \lesssim \sqrt{m/(\eta^2 n)}$. Setting $t=\sqrt{m/(\eta^2 n)}$ and using the limit definition of the exponential function, i.e., $\lim_{t\to\infty}(1-x/t)^t = e^{-x}$, yields for sufficiently small η that (5.7) is bounded by $e^{-c\sqrt{n/m}}$.

We conclude that, with high probability over the initialization, *gradient descent converges with an exponential rate to an arbitrary small empirical risk if the width n is sufficiently large. In addition, the iterates of the descent algorithm even stay in a small fixed neighborhood of the initialization during training.* Because the parameters only move very little, this type of training has also been coined lazy training [COB19].

Similar ideas as above, have led to groundbreaking convergence results of SGD for overparametrized NNs in much more complex and general settings, see, e.g., [DZPS18, LL18, AZLS19].

In the infinite width limit, NN training is practically equivalent to kernel regression, see Subsection 2.1. If we look at Figure 5.3 we see that the most overparametrized NN interpolates the data like a kernel-based interpolator would. In a sense, which was also highlighted in [COB19], this shows that, while overparametrized NNs in the lazy training regime have very nice properties, they essentially act like linear methods.

6 Tangible effects of special architectures

In this section, we describe results that isolate the effects of certain aspects of NN architectures. As we have discussed in Subsection 1.3, typically only either the depth or the number of parameters are used to study theoretical aspects of NNs. We have seen instances of this throughout Sections 3 and 4. Moreover, also in Section 5, we saw that wider NNs enjoy certain very favorable properties from an optimization point of view.

Below, we introduce certain specialized NN architectures. We start with one of the most widely used types of NNs, the *convolutional neural network* (CNN). In Subsection 6.2 we introduce *skip connections* and in Subsection 6.3 we discuss a specific class of CNNs equipped with an encoder-decoder structure that are frequently used in image processing techniques. We introduce the *batch normalization block* in Subsection 6.4. Then, we discuss *sparsely connected* NNs that typically result as an extraction from fully connected NNs in Subsection 6.5. Finally, we briefly comment on recurrent neural networks in Subsection 6.6.

As we have noted repeatedly throughout this manuscript, it is impossible to give a full account of the literature in a short introductory article. In this section, this issue is especially severe since the number of special architectures studied in practice is enormous. Therefore, we had to omit many very influential and widely used neural network architectures. Among those are *graph neural networks*, which handle data from non-Euclidean input spaces. We refer to the survey articles [BBL$^+$17, WPC$^+$21] for a discussion. Another highly successful type of architectures are *(variational) autoencoders* [AHS85, HZ94]. These are neural networks with a bottleneck that enforce a more efficient representation of the data. Similarly, *generative adversarial networks* [GPAM$^+$14] which are composed of two neural networks, one generator and one discriminator, could not be discussed here. Another widely used component of architectures used in practice is the so-called *dropout layer*. This layer functions through removing some neurons randomly during training. This procedure empirically prevents overfitting. An in-detail discussion of the mathematical analysis behind this effect is beyond the scope of this manuscript. We refer to [WZZ$^+$13, SHK$^+$14, HV17, MAV18] instead. Finally, the very successful *attention mechanism* [BCB15, VSP$^+$17], that is the basis of *transformer neural networks*, had to be omitted.

Before we start describing certain effects of special NN architectures, a word of warning is required. The special building blocks, which will be presented below, have been developed based on a specific need in applications and are used and combined in a very flexible way. To describe these tools theoretically without completely inflating the notational load, some simplifying assumptions need to be made. It is very likely that the simplified building blocks do not accurately reflect the practical applications of these tools in all use cases.

6.1 Convolutional neural networks

Especially for very high-dimensional inputs where the input dimensions are spatially related, fully connected NNs seem to require unnecessarily many parameters. For example, in image classification problems, neighboring pixels very often share information and the spatial proximity should be reflected in the architecture. Based on this observation, it appears reasonable to have NNs that have local receptive fields in the sense that they collect information jointly from spatially close inputs. In addition, in image processing, we are not necessarily interested in a universal hypothesis set. A good classifier is invariant under many operations, such as translation or rotation of images. It seems reasonable to hard-code such invariances into the architecture.

These two principles suggest that the receptive field of a NN should be the same on different translated patches of the input. In this sense, parameters of the architecture can be reused. Together, these arguments make up the three fundamental principles of convolutional NNs: local receptive fields, parameter sharing, and equivariant representations, as introduced in [LBD+89]. We will provide a mathematical formulation of convolutional NNs below and then revisit these concepts.

A convolutional NN corresponds to multiple convolutional blocks, which are special types of layers. For a group G, which typically is either $[d] \cong \mathbb{Z}/(d\mathbb{Z})$ or $[d]^2 \cong (\mathbb{Z}/(d\mathbb{Z}))^2$ for $d \in \mathbb{N}$, depending on whether we are performing one-dimensional or two-dimensional convolutions, the convolution of two vectors $a, b \in \mathbb{R}^G$ is defined as

$$(a * b)_i = \sum_{j \in G} a_j b_{j^{-1}i}, \quad i \in G.$$

Now we can define a *convolutional block* as follows: Let $\widetilde{G}$ be a subgroup of G, let $p : G \to \widetilde{G}$ be a so-called *pooling-operator*, and let $C \in \mathbb{N}$ denote the number of channels. Then, for a series of kernels $\kappa_i \in \mathbb{R}^G$, $i \in [C]$, the output of a convolutional block is given by

$$\mathbb{R}^G \ni x \mapsto x' \coloneqq (p(x * \kappa_i))_{i=1}^C \in (\mathbb{R}^{\widetilde{G}})^C. \tag{6.1}$$

A typical example of a pooling operator is for $G = (\mathbb{Z}/(2d\mathbb{Z}))^2$ and $\widetilde{G} = (\mathbb{Z}/(d\mathbb{Z}))^2$ the 2×2 subsampling operator

$$p : \mathbb{R}^G \to \mathbb{R}^{\widetilde{G}}, \quad x \mapsto (x_{2i-1,2j-1})_{i,j=1}^d.$$

Popular alternatives are average pooling or max pooling. These operations then either pass the average or the maximum over patches of similar size. The convolutional kernels correspond to the aforementioned receptive fields. They can be thought of as local if they have small supports, i.e., few nonzero entries.

As explained earlier, a convolutional NN is built by stacking multiple convolutional blocks after another[26]. At some point, the output can be *flattened*, i.e., mapped to a vector and is then fed into a FC NN (see Definition 1.4). We depict this setup in Figure 6.1.

Owing to the fact that convolution is a linear operation, depending on the pooling operation, one may write a convolutional block (6.1) as a FC NN. For example, if $G = (\mathbb{Z}/(2d\mathbb{Z}))^2$ and the 2×2 subsampling pooling operator is used, then the convolutional block could be written as $x \mapsto Wx$ for a block circulant matrix $W \in \mathbb{R}^{(Cd^2) \times (2d)^2}$. Since we require W to have a special structure, we can interpret a convolutional block as a special, restricted feed-forward architecture.

After these considerations, it is natural to ask how the restriction of a NN to a pure convolutional structure, i.e., consisting only of convolutional blocks, will affect the resulting hypothesis set. The first natural question is whether the set of such NNs is still universal in the sense of Theorem 1.15. The answer to this question depends strongly on the type of pooling and convolution that is allowed. If the convolution is performed with padding, then the answer is yes [OS19, Zho20b]. On the other hand, for circular convolutions and without pooling, universality does not hold, but the set of translation equivariant functions can be universally approximated [Yar18b, PV20]. Furthermore, [Yar18b] illuminates the effect of subsample pooling by showing that, if no pooling is applied, then universality cannot be achieved, whereas if pooling is applied

[26] We assume that the definition of a convolutional block is suitably extended to input data in the Cartesian product $(\mathbb{R}^G)^C$. For instance, one can take an affine linear combination of C mappings as in (6.1) acting on each coordinate. Moreover, one may also interject an activation function between the blocks.

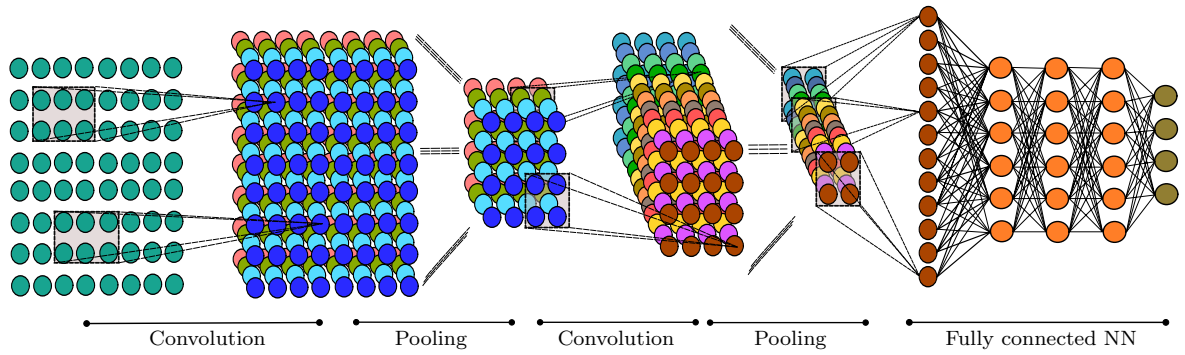


Figure 6.1: Illustration of a convolutional neural network with two-dimensional convolutional blocks and 2×2 subsampling as pooling operation.

then universality is possible. The effect of subsampling in CNNs from the viewpoint of approximation theory is further discussed in [Zho20a]. The role of other types of pooling in enhancing invariances of the hypothesis set will be discussed in Subsection 7.1 below.

6.2 Residual neural networks

Let us first illustrate a potential obstacle when training deep NNs. Consider for $L \in \mathbb{N}$ the product operation

$$\mathbb{R}^L \ni x \mapsto \pi(x) = \prod_{\ell=1}^{L} x_\ell.$$

It is clear that

$$\frac{\partial}{\partial x_k} \pi(x) = \prod_{\ell \neq k}^{L} x_\ell, \quad x \in \mathbb{R}^L.$$

Therefore, for sufficiently large L, we expect that $\left| \frac{\partial \pi}{\partial x_k} \right|$ will be exponentially small, if $|x_\ell| < \lambda < 1$ for all $\ell \in [L]$ or exponentially large, if $|x_\ell| > \lambda > 1$ for all $\ell \in [L]$. The output of a general NN, considered as a directed graph, is found by repeatedly multiplying the input with parameters in every layer along the paths that lead from the input to the output neuron. Due to the aforementioned phenomenon, it is often observed that training NNs suffers from either the exploding or the vanishing gradient problem, which may prevent lower layers from training at all. The presence of an activation function is likely to exacerbate this effect. The exploding or vanishing gradient problem seems to be a serious obstacle towards efficient training of deep NNs.

In addition to the vanishing and exploding gradient problems, there is an empirically observed *degradation problem* [HZRS16]. This phrase describes the phenomenon that FC NNs seem to achieve lower accuracy on both the training and test data when increasing their depth.

From an approximation theoretic perspective, deep NNs should always be superior to shallow NNs. The reason for this is that NNs with two layers can either exactly represent the identity map or approximate it arbitrarily well. Concretely, for the ReLU activation function ϱ_R we have that $x = \varrho_R(x + b) - b$ for $x \in \mathbb{R}^d$ with $x_i > -b_i$, where $b \in \mathbb{R}^d$. In addition, for any activation function ϱ which is continuously differentiable on a neighborhood of some point $\lambda \in \mathbb{R}$ with $\varrho'(\lambda) \neq 0$ one can approximate the identity arbitrary well, see (1.8). Because of this, extending a NN architecture by one layer can only enlarge the associated hypothesis set.

Therefore, one may expect that the degradation problem is more associated with the optimization aspect of learning. This problem is addressed by a small change to the architecture of a feed-forward NN in [HZRS16].

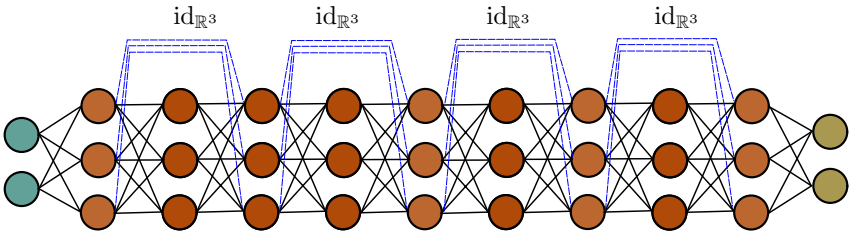


Figure 6.2: Illustration of a neural network with residual blocks.

Instead of defining a FC NN Φ as in (1.1), one can insert a residual block in the ℓ-th layer by redefining[27]

$$\bar{\Phi}^{(\ell)}(x,\theta) = \varrho(\Phi^{(\ell)}(x,\theta)) + \bar{\Phi}^{(\ell-1)}(x,\theta), \tag{6.2}$$

where we assume that $N_\ell = N_{\ell-1}$. Such a block can be viewed as the sum of a regular FC NN and the identity which is referred to as skip connection or *residual connection*. A sketch of a NN with residual blocks is shown in Figure 6.2. Inserting a residual block in all layers leads to a so-called *residual NN*.

A prominent approach to analyze residual NNs is by establishing a connection with optimal control problems and dynamical systems [E17, TvG18, EHL19, LLS19, RH19, LML+20]. Concretely, if each layer of a NN Φ is of the form (6.2), then we have that

$$\bar{\Phi}^{(\ell)} - \bar{\Phi}^{(\ell-1)} = \varrho(\Phi^{(\ell)}) =: h(\ell, \Phi^{(\ell)}),$$

where we abbreviate $\bar{\Phi}^{(\ell)} = \bar{\Phi}^{(\ell)}(x,\theta)$ and set $\bar{\Phi}^{(0)} = x$. Hence, $(\bar{\Phi}^{(\ell)})_{\ell=0}^{L-1}$ corresponds to an Euler discretization of the ODE

$$\dot{\phi}(t) = h(t, \phi(t)), \qquad \phi(0) = x,$$

where $t \in [0, L-1]$ and h is an appropriate function.

Using this relationship, deep residual NNs can be studied in the framework of the well-established theory of dynamical systems, where strong mathematical guarantees can be derived.

6.3 Framelets and U-Nets

One of the most prominent application areas of deep NNs are inverse problems, particularly those in the field of imaging science, see also Subsection 8.1. A specific architectural design of CNNs, namely so-called *U-nets* introduced in [RFB15], seems to perform best for this range of problems. We depict a sketch of a U-net in Figure 6.3. However, a theoretical understanding of the success of this architecture was lacking.

Recently, an innovative approach called *deep convolutional framelets* was suggested in [YHC18], which we now briefly explain. The core idea is to take a frame-theoretic viewpoint, see, e.g., [CKP12], and regard the forward pass of a CNN as a decomposition in terms of a frame (in the sense of a generalized basis). A similar approach will be taken in Subsection 7.2 for understanding the learned kernels using sparse coding. However, based on the analysis and synthesis operators of the corresponding frame, the usage of deep convolutional framelets naturally leads to a theoretical understanding of encoder-decoder architectures, such as U-nets.

Let us describe this approach for one-dimensional convolutions on the group $G := \mathbb{Z}/(d\mathbb{Z})$ with kernels defined on the subgroup $H := \mathbb{Z}/(n\mathbb{Z})$, where $d, n \in \mathbb{N}$ with $n < d$, see also Subsection 6.1. We define the convolution between $u \in \mathbb{R}^G$ and $v \in \mathbb{R}^H$ by zero-padding v, i.e., $g *_\circ v := g * \bar{v}$, where $\bar{v} \in \mathbb{R}^G$ is defined by $\bar{v}_i = v_i$ for $i \in H$ and $\bar{v}_i = 0$ else. As an important tool, we consider the Hankel matrix $\mathbb{H}_n(x) = (x_{i+j})_{i\in G, j\in H} \in \mathbb{R}^{d\times n}$ associated with $x \in \mathbb{R}^G$. As one key property, matrix-vector multiplications with Hankel matrices are translated to convolutions via[28]

$$\langle e^{(i)}, \mathbb{H}_n(x) v\rangle = \sum_{j\in H} x_{i+j} v_j = \langle x, e^{(i)} *_\circ v\rangle, \quad i \in G, \tag{6.3}$$

[27] One can also skip multiple layers, e.g., in [HZRS16] two or three layers skipped, use a simple transformation instead of the identity [SGS15], or randomly drop layers [HSL+16].

[28] Here and in the following we naturally identify elements in $\mathbb{R}^G$ and $\mathbb{R}^H$ with the corresponding vectors in $\mathbb{R}^d$ and $\mathbb{R}^n$.

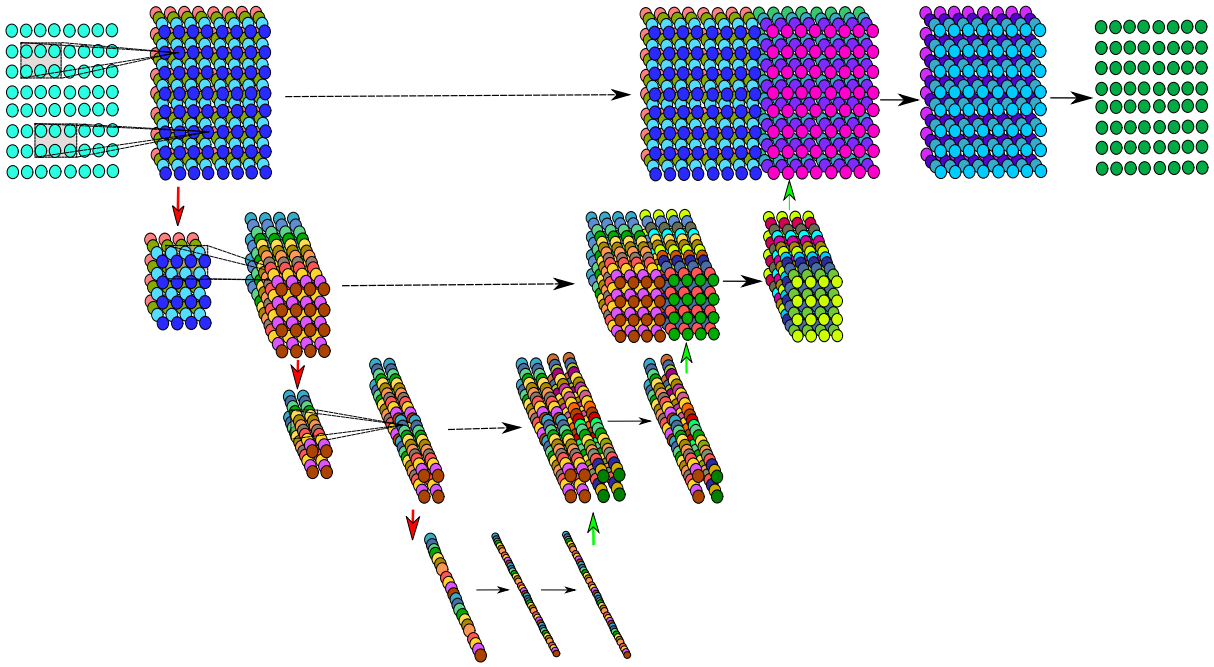


Figure 6.3: Illustration of a simplified U-net neural network. Down-arrows stand for pooling, up arrows for deconvolution or upsampling, right arrows for convolution or fully connected steps. Dashed lines are skip connections.

where $e^{(i)} \coloneqq \mathbb{1}_{\{i\}} \in \mathbb{R}^G$ and $v \in \mathbb{R}^H$, see [YGLD17]. Further, we can recover the k-th coordinate of x by the Frobenius inner product between $\mathbb{H}_n(x)$ and the Hankel matrix associated with $e^{(k)}$, i.e.,

$$\frac{1}{n}\mathrm{Tr}\big(\mathbb{H}_n(e^{(k)})^T\mathbb{H}_n(x)\big) = \frac{1}{n}\sum_{j\in H}\sum_{i\in G} e^{(k)}_{i+j}x_{i+j} = \frac{1}{n}|H|x_k = x_k. \tag{6.4}$$

This allows us to construct global and local bases as follows: Let $p, q \in \mathbb{N}$, let $U = \big[u_1 \cdots u_p\big] \in \mathbb{R}^{d\times p}$, $V = \big[v_1 \cdots v_q\big] \in \mathbb{R}^{n\times q}$, $\widetilde{U} = \big[\tilde{u}_1 \cdots \tilde{u}_p\big] \in \mathbb{R}^{d\times p}$, and $\widetilde{V} = \big[\tilde{v}_1 \cdots \tilde{v}_q\big] \in \mathbb{R}^{n\times q}$, and assume that

$$\mathbb{H}_n(x) = \widetilde{U}U^T\mathbb{H}_n(x)V\widetilde{V}^T. \tag{6.5}$$

For $p \geq d$ and $q \geq n$, this is, for instance, satisfied if U and V constitute frames with $\widetilde{U}$ and $\widetilde{V}$ being their respective dual frames, i.e., $\widetilde{U}U^T = \mathrm{I}_d$ and $V\widetilde{V}^T = \mathrm{I}_n$. As a special case, one can consider orthonormal bases $U = \widetilde{U}$ and $V = \widetilde{V}$ with $p = d$ and $q = n$. In the case $p = q = r \leq n$, where r is the rank of $\mathbb{H}_n(x)$, one can establish (6.5) by choosing the left and right singular vectors of $\mathbb{H}_n(x)$ as $U = \widetilde{U}$ and $V = \widetilde{V}$, respectively. The identity in (6.5), in turn, ensures the following decomposition:

$$x = \frac{1}{n}\sum_{i=1}^{p}\sum_{j=1}^{q}\langle x, u_i *_\circ v_j\rangle \tilde{u}_i *_\circ \tilde{v}_j. \tag{6.6}$$

Observing that the vector $v_j \in \mathbb{R}^H$ interacts locally with $x \in \mathbb{R}^G$ due to the fact that $H \subset G$, whereas $u_i \in \mathbb{R}^G$ acts on the entire vector x, we refer to $(v_j)_{j=1}^q$ as local and $(u_i)_{i=1}^p$ as global bases. In the context of CNNs, v_i can be interpreted as local convolutional kernel and u_i as pooling operation[29]. The proof of (6.6)

[29] Note that $\langle x, u_i *_\circ v_j\rangle$ can also be interpreted as $\langle u_i, v_j \star x\rangle$, where $\star$ denotes the cross-correlation between the zero-padded v_j and x. This is in line with software implementations for deep learning applications, e.g., TensorFlow and PyTorch, where typically cross-correlations are used instead of convolutions.

follows directly from properties (6.3), (6.4), and (6.5) as

$$x_k = \frac{1}{n}\mathrm{Tr}\big(\mathbb{H}_n(e^{(k)})^T\mathbb{H}_n(x)\big) = \frac{1}{n}\mathrm{Tr}\big(\mathbb{H}_n(e^{(k)})^T\widetilde{U}U^T\mathbb{H}_n(x)V\widetilde{V}^T\big) = \frac{1}{n}\sum_{i=1}^{p}\sum_{j=1}^{q}\langle u_i, \mathbb{H}_n(x)v_j\rangle\langle\tilde{u}_i, \mathbb{H}_n(e^{(k)})\tilde{v}_j\rangle.$$

The decomposition in (6.6) can now be interpreted as a composition of an encoder and a decoder,

$$x \mapsto C = (\langle x, u_i *_\circ v_j\rangle)_{i\in[p],j\in[q]} \quad \text{and} \qquad C \mapsto \frac{1}{n}\sum_{i=1}^{p}\sum_{j=1}^{q} C_{i,j}\tilde{u}_i *_\circ \tilde{v}_j, \tag{6.7}$$

which relates it to CNNs equipped with an encoder-decoder structure such as U-nets, see Figure 6.3. Generalizing this approach to multiple channels, it is possible to stack such encoders and decoders which leads to a layered version of (6.6). In [YHC18] it is shown that one can make an informed decision on the number of layers based on the rank of $\mathbb{H}_n(x)$, i.e., the complexity of the input features x. Moreover, also an activation function such as the ReLU or bias vectors can be included. The key question one can then ask is how the kernels can be chosen to obtain sparse coefficients C in (6.7) and a decomposition such as (6.6), i.e., perfect reconstruction. If U and V are chosen as the left and right singular vectors of $\mathbb{H}_n(x)$, one obtains a very sparse, however input-dependent, representation in (6.6) due to the fact that

$$C_{i,j} = \langle x, u_i *_\circ v_j\rangle = \langle u_i, \mathbb{H}_n(x)v_j\rangle = 0, \quad i \neq j.$$

Finally, using the framework of deep convolutional framelets, theoretical reasons for including skip connections can be derived, since they aid to obtain a perfect reconstruction.

6.4 Batch normalization

Batch normalization is a building block of NNs that was invented in [IS15] with the goal to reduce so-called *internal covariance shift*. In essence, this phrase describes the (undesirable) situation where during training each layer receives inputs with different distribution. A batch normalization block is defined as follows: For points $b = (y^{(i)})_{i=1}^m \in (\mathbb{R}^n)^m$ and $\beta, \gamma \in \mathbb{R}$, we define

$$\mathrm{BN}_b^{(\beta,\gamma)}(y) \coloneqq \gamma\,\frac{y-\mu_b}{\sigma_b} + \beta, \quad y \in \mathbb{R}^n, \quad \text{with} \quad \mu_b = \frac{1}{m}\sum_{i=1}^{m} y^{(i)} \quad \text{and} \quad \sigma_b^2 = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)}-\mu_b)^2, \tag{6.8}$$

where all operations are to be understood componentwise, see Figure 6.4.

Such a batch normalization block can be added into a NN architecture. Then b is the output of the previous layer over a batch or the whole training data[30]. Furthermore, the parameters β, γ are variable and can be learned during training. Note that, if one sets $\beta = \mu_b$ and $\gamma = \sigma_b$, then $\mathrm{BN}_b^{(\beta,\gamma)}(y) = y$ for all $y \in \mathbb{R}^n$. Therefore, a batch normalization block does not negatively affect the expressivity of the architecture. On the other hand, batch normalization does have a tangible effect on the optimization aspects of deep learning. Indeed, in [STIM18, Theorem 4.1], the following observation was made:

[30] In practice, one typically uses a moving average to estimate the mean μ and the standard deviation σ of the output of the previous layer over the whole training data by only using batches.

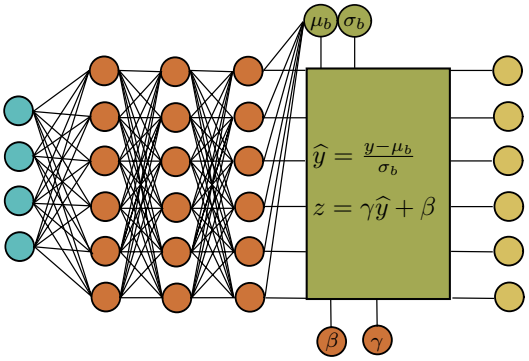


Figure 6.4: A batch normalization block after a fully connected neural network. The parameters μ_b, σ_b are the mean and the standard deviation of the output of the fully connected network computed over a batch s, i.e., a set of inputs. The parameters β, γ are learnable parts of the batch normalization block.

Proposition 6.1 (Smoothening effect of batch normalization)**.** *Let $m \in \mathbb{N}$ with $m \geq 2$ and for every $\beta, \gamma \in \mathbb{R}$ define $\mathcal{B}^{(\beta,\gamma)} \colon \mathbb{R}^m \to \mathbb{R}^m$ by*

$$\mathcal{B}^{(\beta,\gamma)}(b) = (\mathrm{BN}_b^{(\beta,\gamma)}(y^{(1)}), \dots, \mathrm{BN}_b^{(\beta,\gamma)}(y^{(m)})), \quad b = (y^{(i)})_{i=1}^m \in \mathbb{R}^m,$$

where $\mathrm{BN}_b^{(\beta,\gamma)}$ is given as in (6.8). Let $\beta, \gamma \in \mathbb{R}$ and let $r \colon \mathbb{R}^m \to \mathbb{R}$ be a differentiable function. Then it holds for every $b \in \mathbb{R}^m$ that

$$\|\nabla(r \circ \mathcal{B}^{(\beta,\gamma)})(b)\|_2^2 = \frac{\gamma^2}{\sigma_b^2}\big(\|\nabla r(b)\|^2 - \frac{1}{m}\langle \mathbf{1}, \nabla r(b)\rangle^2 - \frac{1}{m}\langle \mathcal{B}^{(0,1)}(b), \nabla r(b)\rangle^2\big),$$

where $\mathbf{1} = (1, \dots, 1) \in \mathbb{R}^m$ and σ_b^2 is given as in (6.8).

For multi-dimensional $y^{(i)} \in \mathbb{R}^n$, $i \in [m]$, the same statement holds for all components as, by definition, the batch normalization block acts componentwise. Proposition 6.1 follows from a convenient representation of the Jacobian of the mapping $\mathcal{B}^{(\beta,\gamma)}$, given by

$$\frac{\partial \mathcal{B}^{(\beta,\gamma)}(b)}{\partial b} = \frac{\gamma}{\sigma_b}\Big(\mathrm{I}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^T - \frac{1}{m}\mathcal{B}^{(0,1)}(b)(\mathcal{B}^{(0,1)}(b))^T\Big), \quad b \in \mathbb{R}^m,$$

and the fact that $\{\frac{\mathbf{1}}{\sqrt{m}}, \frac{1}{\sqrt{m}}\mathcal{B}^{(0,1)}(b)\}$ constitutes an orthonormal set.

Choosing r to mimic the empirical risk of a learning task, Proposition 6.1 shows that, in certain situations—for instance, if γ is smaller than σ_b or if m is not too large—a batch normalization block can considerably reduce the magnitude of the derivative of the empirical risk with respect to the input of the batch normalization block. By the chain rule, this implies that also the derivative of the empirical risk with respect to NN parameters influencing the input of the batch normalization block is reduced.

Interestingly, a similar result holds for second derivatives [STIM18, Theorem 4.2] if r is twice differentiable. One can conclude that adding a batch normalization block increases the smoothness of the optimization problem. Since the parameters β and γ were introduced, including a batch normalization block also increases the dimension of the optimization problem by two.

6.5 Sparse neural networks and pruning

For deep FC NNs, the number of trainable parameters usually scales like the square of the number of neurons. For reasons of computational complexity and memory efficiency, it appears sensible to seek for techniques to reduce the number of parameters or extract *sparse subnetworks* (see Figure 6.5) without affecting the output

of a NN much. One way to do this is by *pruning* [LDS89, HMD16]. Here, certain parameters of a NN are removed after training. This is done, for example, by setting these parameters to zero.

In this context, the *lottery ticket hypothesis* was formulated in [FC18]. It states: "A randomly-initialized, dense NN contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original NN after training for at most the same number of iterations". In [RWK+20] a similar hypothesis was made and empirically studied. There, it is claimed that, for a sufficiently overparametrized NN, there exists a subnetwork that matches the performance of the large NN after training without being trained itself, i.e., already at initialization.

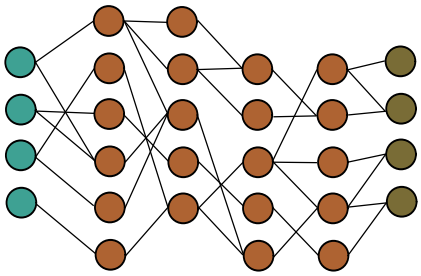


Figure 6.5: A neural network with sparse connections.

Under certain simplifying assumptions, the existence of favorable subnetworks is quite easy to prove. We can use a technique that was previously indirectly used in Subsection 4.2—the Carathéodory Lemma. This result states the following: Let $n \in \mathbb{N}$, $C \in (0, \infty)$, and let $(\mathcal{H}, \|\cdot\|)$ be a Hilbert space. Let $F \subset \mathcal{H}$ with $\sup_{f \in F} \|f\| \leq C$ and let $g \in \mathcal{H}$ be in the convex hull of F. Then there exist $f_i \in F$, $i \in [n]$, and $c \in [0,1]^n$ with $\|c\|_1 = 1$ such that

$$\left\| g - \sum_{i=1}^{n} c_i f_i \right\| \leq \frac{C}{\sqrt{n}},$$

see, e.g., [Ver18, Theorem 0.0.2].

Proposition 6.2 (Carathéodory pruning)**.** *Let $d, n \in \mathbb{N}$, with $n \geq 100$ and let μ be a probability measure on the unit ball $B_1(0) \subset \mathbb{R}^d$. Let $a = ((d, n, 1), \varrho_R)$ be the architecture of a two-layer ReLU network and let $\theta \in \mathbb{R}^{P((d,n,1))}$ be corresponding parameters such that*

$$\Phi_a(\cdot, \theta) = \sum_{i=1}^{n} w_i^{(2)} \varrho_R(\langle (w_i^{(1)}, \cdot\rangle + b_i^{(1)})),$$

where $(w_i^{(1)}, b_i^{(1)}) \in \mathbb{R}^d \times \mathbb{R}$, $i \in [n]$, and $w^{(2)} \in \mathbb{R}^n$. Assume that for every $i \in [n]$ it holds that $\|w_i^{(1)}\|_2 \leq 1/2$ and $b_i^{(1)} \leq 1/2$. Then there exists a parameter $\tilde{\theta} \in \mathbb{R}^{P((d,n,1))}$ with at least 99% of its entries being zero such that

$$\|\Phi_a(\cdot, \theta) - \Phi_a(\cdot, \tilde{\theta})\|_{L^2(\mu)} \leq \frac{15 \|w^{(2)}\|_1}{\sqrt{n}}.$$

Specifically, there exists an index set $I \subset [n]$ with $|I| \leq n/100$ such that $\tilde{\theta}$ satisfies that

$$\widetilde{w}_i^{(2)} = 0, \quad \text{if } i \notin I, \qquad \text{and} \qquad (\widetilde{w}_i^{(1)}, \tilde{b}_i^{(1)}) = \begin{cases} (w_i^{(1)}, b_i^{(1)}), & \text{if } i \in I, \\ (0,0), & \text{if } i \notin I. \end{cases}$$

The result is clear if $w^{(2)} = 0$. Otherwise, define

$$f_i \coloneqq \|w^{(2)}\|_1 \varrho_R(\langle w_i^{(1)}, \cdot\rangle + b_i^{(1)}), \quad i \in [n],$$

and observe that $\Phi_a(\cdot, \theta)$ is in the convex hull of $\{f_i\}_{i=1}^n \cup \{-f_i\}_{i=1}^n$. Moreover, by the Cauchy–Schwarz inequality, it holds that

$$\|f_i\|_{L^2(\mu)} \leq \|w^{(2)}\|_1 \|f_i\|_{L^\infty(B_1(0))} \leq \|w^{(2)}\|_1.$$

We conclude with the Carathéodory Lemma that there exists $I \subset [n]$ with $|I| = \lfloor n/100 \rfloor \geq n/200$ and $c_i \in [-1, 1]$, $i \in I$, such that

$$\left\| \Phi_a(\cdot, \theta) - \sum_{i \in I} c_i f_i \right\|_{L^2(\mu)} \leq \frac{\|w^{(2)}\|_1}{\sqrt{|I|}} \leq \frac{\sqrt{200} \|w^{(2)}\|_1}{\sqrt{n}},$$

which yields the result.

Proposition 6.2 shows that certain, very wide NNs can be approximated very well by sparse subnetworks where only the output weight matrix needs to be changed. The argument of Proposition 6.2 is inspired by [BK18], where a much more refined result is shown for deep NNs.

6.6 Recurrent neural networks

Recurrent NNs are NNs where the underlying graph is allowed to exhibit cycles as in Figure 6.6, see [Hop82, RHW86, Elm90, Jor90]. Previously, we had excluded cyclic computational graphs. For a feed-forward NN, the computation of internal states is naturally performed step by step through the layers. Since the output of a layer does not affect previous layers, the order in which the computations of the NN are performed corresponds to the order of the layers. For recurrent NNs, the concept of layers does not exist, and the order of operations is much more delicate. Therefore, one considers time steps. In each time step, all possible computations of the graph are applied to the current state of the NN. This yields a new internal state. Given that time steps arise naturally from the definition of recurrent NNs, this NN type is typically used for sequential data.

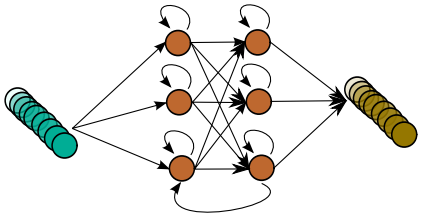


Figure 6.6: Sketch of a recurrent neural network. Cycles in the computational graph incorporate the sequential structure of the input and output.

If the input to a recurrent NN is a sequence, then every input determines the internal state of the recurrent NN for the following inputs. Therefore, one can claim that these NNs exhibit a memory. This fact is extremely desirable in natural language processing, which is why recurrent NNs are widely used in this application.

Recurrent NNs can be trained similarly to regular feed-forward NNs by an algorithm called *backpropagation through time* [MP69, Wer88, WZ95]. This procedure essentially unfolds the recurrent structure yielding a classical NN structure. However, the algorithm may lead to very deep structures. Due to the vanishing and exploding gradient problem discussed earlier, very deep NNs are often hard to train. Because of this, special recurrent structures were introduced that include gates which prohibit too many recurrent steps; these include the widely used LSTMs [HS97].

The application area of recurrent NNs is typically quite different from that of regular NNs since they are specialized on sequential data. Therefore, it is hard to quantify the effect of a recurrent connection on a fully connected NN. However, it is certainly true that with recurrent connections certain computations can be performed much more efficiently than with feed-forward NN structures. A particularly interesting construction can be found in [BF19, Theorem 4.4], where it is shown that a fixed size, recurrent NN with ReLU activation function, can approximate the function $x \mapsto x^2$ to any desired accuracy. The reason for this efficient representation can be seen when considering the self-referential definition of the approximant to $x - x^2$ shown in Figure 3.2.

On the other hand, with feed-forward NNs, it transpires from Theorem 3.3 that the approximation error of fixed-sized ReLU NNs for any non-affine function is greater than a positive lower bound.

7 Describing the features a deep neural network learns

This section presents two viewpoints which help in understanding the nature of features that can be described by NNs. Section 7.1 summarizes aspects of the so-called *scattering transform* which constitutes a specific NN architecture that can be shown to satisfy desirable properties, such as translation and deformation invariance. Section 7.2 relates NN features to the current paradigm of *sparse coding*.

7.1 Invariances and the scattering transform

One of the first theoretical contributions to the understanding of the mathematical properties of CNNs is [Mal12]. The approach taken in that work is to consider specific CNN architectures with *fixed* parameters

that result in a stand-alone feature descriptor whose output may be fed into a subsequent classifier (for example, a kernel support vector machine or a trainable FC NN). From an abstract point of view, a feature descriptor is a function Ψ mapping from a signal space, such as $L^2(\mathbb{R}^d)$ or the space of piecewise smooth functions, to a feature space. In an ideal world, such a classifier should "factor" out invariances that are irrelevant to a subsequent classification problem while preserving all other information of the signal. A very simple example of a classifier which is invariant under translations is the Fourier modulus $\Psi\colon L^2(\mathbb{R}^d) \to L^2(\mathbb{R}^d)$, $u \mapsto |\hat{u}|$. This follows from the fact that a translation of a signal u results in a modulation of its Fourier transform, i.e., $\widehat{u(\cdot - \tau)}(\omega) = e^{-2\pi i \langle \tau, \omega \rangle} \hat{u}(\omega)$, $\tau, \omega \in \mathbb{R}^d$. Furthermore, in most cases (for example, if u is a generic compactly supported function [GKR20]), u can be reconstructed up to a translation from its Fourier modulus [GKR20] and an energy conservation property of the form $\|\Psi(u)\|_{L^2} = \|u\|_{L^2}$ holds true. Translation invariance is, for example, typically exhibited by image classifiers, where the label of an image does not change if it is translated.

In practical problems many more invariances arise. Providing an analogous representation that factors out general invariances would lead to a significant reduction in the problem dimensionality and constitutes an extremely promising route towards dealing with the very high dimensionality that is commonly encountered in practical problems [Mal16]. This program is carried out in [Mal12] for additional invariances with respect to deformations $u \mapsto u_\tau := u(\cdot - \tau(\cdot))$, where $\tau\colon \mathbb{R}^d \to \mathbb{R}^d$ is a smooth mapping. Such transformations may occur in practice, for instance, as image warpings. In particular, a feature descriptor Ψ is designed that, with a suitable norm $\|\cdot\|$ on the image of Ψ,

(a) is Lipschitz continuous with respect to deformations in the sense that $\|\Psi(u) - \Psi(u_\tau)\| \lesssim K(\tau, \nabla\tau, \nabla^2\tau)$ holds for some K that only mildly depends on τ and essentially grows linearly in $\nabla\tau$ and $\nabla^2\tau$,

(b) is almost (i.e., up to a small and controllable error) invariant under translations of the input data, and

(c) contains all relevant information on the input data in the sense that an energy conservation property

$$\|\Psi(u)\| \approx \|u\|_{L^2}$$

holds true.

Observe that, while the action of translations only represents a d-parameter group, the action of deformations/warpings represents an infinite-dimensional group. Hence, a deformation invariant feature descriptor represents a big potential for dimensionality reduction. Roughly speaking, the feature descriptor Ψ of [Mal12] (also coined the *scattering transform*) is defined by collecting features that are computed by iteratively applying a wavelet transform followed by a pointwise modulus non-linearity and a subsequent low-pass filtering step, i.e.,

$$|||u * \psi_{j_1}| * \psi_{j_2} * \ldots| * \psi_{j_\ell}| * \varphi_J,$$

where ψ_j refers to a wavelet at scale j and φ_J refers to a scaling function at scale J. The collection of all these so-called *scattering coefficients* can then be shown to satisfy the properties in (a)–(c) above in a suitable (asymptotic) sense. The proof of this result relies on a subtle interplay between a "deformation covariance" property of the wavelet transform and a "regularizing" property of the operation of convolution with the modulus of a wavelet. We remark that similar results can be shown also for different systems, such as Gabor frames [WGB17, CL19].

7.2 Hierarchical sparse representations

The previous approach modeled the learned features by a specific dictionary, namely wavelets. It is well known that one of the striking properties of wavelets is to provide sparse representations for functions belonging to certain function classes. More generally, we speak of sparse representations with respect to a representation system. For a vector $x \in \mathbb{R}^d$, a sparsifying representation system $D \in \mathbb{R}^{d \times p}$—also called *dictionary*—is such that $x = D\phi$ with the coefficients $\phi \in \mathbb{R}^p$ being sparse in the sense that $\|\phi\|_0 := |\operatorname{supp}(\phi)| = |\{i \in [p]\colon \phi_i \neq 0\}|$ is small compared to p. A similar definition can be made for signals in infinite-dimensional spaces. Taking

sparse representations into account, the theory of sparse coding provides an approach to a theoretical understanding of the features a deep NN learns.

One common method in image processing is the utilization of not the entire image but overlapping patches of it, coined *patch-based image processing*. Thus of particular interest are local dictionaries which sparsify those patches, but presumably not the global image. This led to the introduction of the *convolutional sparse coding* model (CSC model), which links such local and global behaviors. Let us describe this model for one-dimensional convolutions on the group $G := \mathbb{Z}/(d\mathbb{Z})$ with kernels supported on the subgroup $H := \mathbb{Z}/(n\mathbb{Z})$, where $d, n \in \mathbb{N}$ with $n < d$, see also Subsection 6.1. The corresponding CSC model is based on a decomposition of a global signal $x \in (\mathbb{R}^G)^c$ with $c \in \mathbb{N}$ channels as

$$x_i = \sum_{j=1}^{C} \kappa_{i,j} * \phi_j, \quad i \in [c], \tag{7.1}$$

where $\phi \in (\mathbb{R}^G)^C$ is supposed to be a sparse representation with $C \in \mathbb{N}$ channels and $\kappa_{i,j} \in \mathbb{R}^G$, $i \in [c]$, $j \in [C]$, are local kernels with $\operatorname{supp}(\kappa_{i,j}) \subset H$. Let us consider a patch $((x_i)_{g+h})_{i\in[c],h\in H}$ of n adjacent entries, starting at position $g \in G$, in each channel of x. The condition on the support of the kernels $\kappa_{i,j}$ and the representation in (7.1) imply that this patch is only affected by a stripe of at most $(2n-1)$ entries in each channel of ϕ. The local, patch-based sparsity of the representation ϕ can thus be appropriately measured via

$$\|\phi\|_{0,\infty}^{(n)} := \max_{g\in G} \|((\phi_j)_{g+k})_{j\in[C],k\in[2n-1]}\|_0,$$

see [PSE17]. Furthermore, note that we can naturally identify x and ϕ with vectors in $\mathbb{R}^{dc}$ and $\mathbb{R}^{dC}$ and write $x = D\phi$, where $D \in \mathbb{R}^{dc\times dC}$ is a matrix consisting of circulant blocks, typically referred to as a *convolutional dictionary*.

The relation between the CSC model and deep NNs is revealed by applying the CSC model in a layer-wise fashion [PRE17, SPRE18, PRSE18]. To see this, let $C_0 \in \mathbb{N}$ and for every $\ell \in [L]$ let $C_\ell, k_\ell \in \mathbb{N}$ and let $D^{(\ell)} \in \mathbb{R}^{dC_{\ell-1}\times dC_\ell}$ be a convolutional dictionary with kernels supported on $\mathbb{Z}/(n_\ell\mathbb{Z})$. A signal $x = \phi^{(0)} \in \mathbb{R}^{dC_0}$ is said to belong to the corresponding *multi-layered CSC model* (ML-CSC model) if there exist coefficients $\phi^{(\ell)} \in \mathbb{R}^{dC_\ell}$ with

$$\phi^{(\ell-1)} = D^{(\ell)}\phi^{(\ell)} \quad \text{and} \quad \|\phi^{(\ell)}\|_{0,\infty}^{(n_\ell)} \le k_\ell, \quad \ell \in [L]. \tag{7.2}$$

We now consider the problem of reconstructing the sparse coefficients $(\phi^{(\ell)})_{\ell=1}^L$ from a noisy signal $\tilde{x} := x + \nu$, where the noise $\nu \in \mathbb{R}^{dC_0}$ is assumed to have small ℓ^2-norm and x is assumed to follow the ML-CSC model in (7.2). In general, this problem is NP-hard. However, under suitable conditions on the ML-CSC model, it can be approximately solved, for instance, by a layered thresholding algorithm.

More precisely, for $D \in \mathbb{R}^{dc\times dC}$ and $b \in \mathbb{R}^{dC}$, we define a *soft-thresholding operator* by

$$\mathcal{T}_{D,b}(x) := \varrho_R(D^T x - b) - \varrho_R(-D^T x - b), \quad x \in \mathbb{R}^{dc}, \tag{7.3}$$

where $\varrho_R(x) = \max\{0, x\}$ is applied componentwise. If $x = D\phi$ as in (7.1), we obtain $\phi \approx \mathcal{T}_{D,b}(x)$ roughly under the following conditions: The distance of ϕ and $\psi := D^T x = D^T D\phi$ can be bounded using the local sparsity of ϕ and the mutual coherence and locality of the kernels of the convolutional dictionary D. For a suitable threshold b, the mapping $\psi \mapsto \varrho_R(\psi - b) - \varrho_R(-\psi - b)$ further recovers the support of ϕ by nullifying entries of ψ with $\psi_i \le |b_i|$. Utilizing the soft-thresholding operator (7.3) iteratively for corresponding vectors $b^{(\ell)} \in \mathbb{R}^{dC_\ell}$, $\ell \in [L]$, then suggests the following approximations:

$$\phi^{(\ell)} \approx (\mathcal{T}_{D^{(\ell)},b^{(\ell)}} \circ \cdots \circ \mathcal{T}_{D^{(1)},b^{(1)}})(\tilde{x}), \quad \ell \in [L].$$

The resemblance with the realization of a CNN with ReLU activation function is evident. The transposed dictionary $(D^{(\ell)})^T$ can be regarded as modeling the learned convolutional kernels, the threshold $b^{(\ell)}$ models the bias vector, and the soft-thresholding operator $\mathcal{T}_{D^{(\ell)},b^{(\ell)}}$ mimics the application of a convolutional block with a ReLU non-linearity in the ℓ-th layer.

Using this model, a theoretical understanding of CNNs from the perspective of sparse coding is now at hand. This novel perspective gives a precise mathematical meaning of the kernels in a CNN as sparsifying dictionaries of an ML-CSC model. Moreover, the forward pass of a CNN can be understood as a layered thresholding algorithm for decomposing a noisy signal $\tilde{x}$. The results derived are then of the following flavor: Given a suitable reconstruction procedure such as thresholding or ℓ_1-minimization, the sparse coefficients $(\phi^{(\ell)})_{\ell=1}^{L}$ of a signal x following a ML-CSC model can be stably recovered from the noisy signal $\tilde{x}$ under certain hypotheses on the ingredients of the ML-CSC model.

8 Effectiveness in natural sciences

The theoretical insights of the previous sections do not always accurately describe the performance of NNs in applications. Indeed, there often exists a considerable gap between the predictions of approximation theory and the practical performance of NNs [AD20].

In this section, we consider concrete applications which have been very successfully solved with deep-learning-based methods. In Section 8.1 we present an overview of deep-learning-based algorithms applied to inverse problems. Section 8.2 then continues by describing how NNs can be used as a numerical ansatz for solving PDEs, highlighting their use in the solution of the multi-electron Schrödinger equation.

8.1 Deep neural networks meet inverse problems

The area of inverse problems, predominantly in imaging, was presumably the first class of mathematical methods embracing deep learning with overwhelming success. Let us consider a forward operator $K\colon \mathcal{Y} \to \mathcal{X}$ with $\mathcal{X}, \mathcal{Y}$ being Hilbert spaces and the associated inverse problem of finding $y \in \mathcal{Y}$ such that $Ky = x$ for given features $x \in \mathcal{X}$. The classical model-based approach to regularization aims to approximate K by invertible operators, and is hence strongly based on functional analytic principles. Today, such approaches take well-posedness of the approximation, convergence properties, as well as the structure of regularized solutions into account. The last item allows to incorporate prior information of the original solution such as regularity, sharpness of edges, or—in the case of sparse regularization [JMS17]—a sparse coefficient sequence with respect to a prescribed representation system. Such approaches are typically realized in a variational setting and hence aim to minimize functionals of the form

$$\|Ky - x\|^2 + \alpha R(y),$$

where $\alpha \in (0, \infty)$ is a regularization parameter, $R\colon \mathcal{Y} \to [0, \infty)$ a regularization term, and $\|\cdot\|$ denotes the norm on $\mathcal{Y}$. As said, the regularization term aims to model structural information about the desired solution. However, one main hurdle in this approach is the problem that typically solution classes such as images from computed tomography cannot be modeled accurately enough to, for instance, allow reconstruction under the constraint of a significant amount of missing features.

This has opened the door to data-driven approaches, and recently, deep NNs. Solvers of inverse problems which are based on deep learning techniques can be roughly categorized into three classes:

1. *Supervised approaches:* The most straightforward approach is to train a NN $\Phi(\cdot, \theta)\colon \mathcal{X} \to \mathcal{Y}$ end-to-end, i.e., to completely learn the map from data x to the solution y. More advanced approaches in this direction incorporate information about the operator K into the NN such as in [AÖ17, GOW19, MLE21]. Yet another type of approaches aims to combine deep NNs with classical model-based approaches. The first suggestion in this realm was to start by applying a standard solver, followed by a deep NN $\Phi(\cdot, \theta)\colon \mathcal{Y} \to \mathcal{Y}$ which serves as a denoiser for specific reconstruction artifacts, e.g., [JMFU17]. This was followed by more sophisticated methods such as plug-and-play frameworks for coupling inversion and denoising [REM17].

2. *Semi-supervised approaches:* These type of approaches aim to encode the regularization by a deep NN $\Phi(\cdot, \theta)\colon \mathcal{Y} \to [0, \infty)$. The underlying idea is often to require stronger regularization on solutions $y^{(i)}$

that are more prone to artifacts or other effects of the instability of the problem. On solutions where typically few artifacts are observed less regularization can be used. Therefore, the learning algorithm only requires a set of labels $(y^{(i)})_{i=1}^m$ as well as a method to assess how hard the inverse problem for this label would be. In this sense, the algorithm can be considered semi-supervised. This idea was followed, for example, in [LÖS18, LSAH20]. Taking a Bayesian viewpoint, one can also learn prior distributions as deep NNs, which was done in [BZAJ20].

3. *Unsupervised approaches:* One highlight of what we might coin unsupervised approaches in our problem setting is the introduction of deep image priors in [DKMB20, UVL18]. The key idea is to parametrize the solutions y as the output of a NN $\Phi(\xi, \cdot)\colon \mathcal{P} \to \mathcal{Y}$ with parameters in a suitable space $\mathcal{P}$, applied to a fixed input ξ. Then, for given features x, one tries to solve $\min_{\theta\in\mathcal{P}} \|K\Phi(\xi,\theta) - x\|^2$ in order to obtain parameters $\hat{\theta} \in \mathcal{P}$ that yield a solution candidate $y = \Phi(\xi, \hat{\theta})$. Here often early stopping is applied in the training of the network parameters.

As can be seen, one key conceptual question is how to "take the best out of both worlds", in the sense of optimally combining classical (model-based) methods—in particular the forward operator K—with deep learning. This is certainly sensitively linked to all characteristics of the particular application at hand, such as availability and accuracy of training data, properties of the forward operator, or requirements for the solution. And each of the three classes of hybrid solvers follows a different strategy.

Let us now discuss advantages and disadvantages of methods from the three categories with a particular focus on a mathematical foundation. *Supervised* approaches suffer on the one hand from the problem that often ground-truth data is not available or only in a very distorted form, leading to the fact that synthetic data constitutes a significant part of the training data. Thus the learned NN will mainly perform as well as the algorithm which generated the data, but not significantly improve it—only from an efficiency viewpoint. On the other hand, the inversion is often highly ill-posed, i.e., the inversion map has a large Lipschitz constant, which negatively affects the generalization ability of the NN. Improved approaches incorporate knowledge about the forward operator K as discussed, which helps to circumvent this issue.

One significant advantage of *semi-supervised* approaches is that the underlying mathematical model of the inverse problem is merely augmented by the neural network-based regularization. Assuming that the learned regularizer satisfies natural assumptions, convergence proofs or stability estimates for the resulting regularized methods are still available.

Finally, *unsupervised* approaches have the advantage that the regularization is then fully due to the specific architecture of the deep NN. This makes these methods slightly easier to understand theoretically, although, for instance, the deep prior approach in its full generality is still lacking a profound mathematical analysis.

8.2 PDE-based models

Besides applications in image processing and artificial intelligence, deep learning methods have recently strongly impacted the field of numerical analysis. In particular, regarding the numerical solution of high-dimensional PDEs. These PDEs are widely used as a model for complex processes and their numerical solution presents one of the biggest challenges in scientific computing. We mention three exemplary problem classes:

1. *Black–Scholes model:* The Nobel award-winning theory of Fischer Black, Robert Merton, and Myron Scholes proposes a linear PDE model for the determination of a fair price of a (complex) financial derivative. The dimensionality of the model corresponds to the number of financial assets which is typically quite large. The classical linear model, which can be solved efficiently via Monte Carlo methods is quite limited. In order to take into account more realistic phenomena such as default risk, the PDE that models a fair price becomes nonlinear, and much more challenging to solve. In particular (with the notable exception of Multilevel Picard algorithms [EHJK19]) no general algorithm exists that provably scales well with the dimension.

2. *Schrödinger equation:* The electronic Schrödinger equation describes the stationary nonrelativistic behavior of a quantum mechanical electron system in the electric field generated by the nuclei of a molecule. Its numerical solution is required to obtain stable molecular configurations, compute vibrational spectra, or obtain forces governing molecular dynamics. If the number of electrons is large, this is again a high-dimensional problem and to date there exist no satisfactory algorithms for its solution: It is well known that different gold standard methods may produce completely different energy predictions, for example, when applied to large delocalized molecules, rendering these methods useless for those problems.

3. *Hamilton–Jacobi–Bellman equation:* The Hamilton–Jacobi–Bellman (HJB) equation models the value function of (deterministic or stochastic) optimal control problems. The underlying dimensionality of the model corresponds to the dimension of the space of states to be controlled and tends to be rather high in realistic applications. The high dimensionality, together with the fact that HJB equations typically tend to be fully nonlinear with non-smooth solutions, renders the numerical solution of HJB equations extremely challenging and no general algorithms exist for this problem.

Due to the favorable approximation results of NNs for high-dimensional functions (see especially Subsection 4.3), it might not come as a surprise that a NN ansatz has proven to be quite successful in solving the aforementioned PDE models. A pioneering work in this direction is [HJE18] which uses the backwards SDE reformulation of semilinear parabolic PDEs to reformulate the evaluation of such a PDE at a specific point as an optimization problem that can be solved by the deep learning paradigm. The resulting algorithm proves quite successful in the high-dimensional regime and, for instance, enables the efficient modeling of complex financial derivatives including nonlinear effects such as default risk. Another approach specifically tailored to the numerical solution of HJB equations is [NZGK21]. In this work, one uses the Pontryagin principle to generate samples of the PDE solution along solutions of the corresponding boundary value problem. Other numerical approaches include the *Deep Ritz Method* [EY18], where a Dirichlet energy is minimized over a set of NNs, or so-called *Physics Informed Neural Networks* [RPK19], where typically the PDE residual is minimized along with some natural constraints, for instance, to enforce boundary conditions.

Deep-learning-based methods arguably work best if they are combined with domain knowledge to inspire NN architecture choices. We would like to illustrate this interplay at the hand of a specific and extremely relevant example: the electronic Schrödinger equation (under the Born–Oppenheimer approximation) which amounts to finding the smallest nonzero eigenvalue of the eigenvalue problem

$$\mathcal{H}_R\psi = \lambda_\psi\psi, \tag{8.1}$$

for $\psi\colon \mathbb{R}^{3\times n} \to \mathbb{R}$, where the Hamiltonian

$$(\mathcal{H}_R\psi)(r) = -\sum_{i=1}^{n}\frac{1}{2}(\Delta_{r_i}\psi)(r) - \left(\sum_{i=1}^{n}\sum_{j=1}^{p}\frac{Z_j}{\|r_i - R_j\|_2} - \sum_{i=1}^{p-1}\sum_{j=i+1}^{p}\frac{Z_iZ_j}{\|R_i - R_j\|_2} - \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{\|r_i - r_j\|_2}\right)\psi(r)$$

describes the kinetic energy (first term) as well as Coulomb attraction force between electrons and nuclei (second and third term) and the Coulomb repulsion force between different electrons (third term). Here, the coordinates $R = \big[R_1 \dots R_p\big] \in \mathbb{R}^{3\times p}$ refer to the positions of the nuclei, $(Z_i)_{i=1}^p \in \mathbb{N}^p$ denote the atomic numbers of the nuclei, and the coordinates $r = \big[r_1, \dots, r_n\big] \in \mathbb{R}^{3\times n}$ refer to the positions of the electrons. The associated eigenfunction ψ describes the so-called *wavefunction* which can be interpreted in the sense that $|\psi(r)|^2/\|\psi\|_{L^2}^2$ describes the joint probability density of the n electrons to be located at r. The smallest solution λ_ψ of (8.1) describes the *ground state energy* associated with the nuclear coordinates R. It is of particular interest to know the ground state energy for all nuclear coordinates, the so-called *potential energy surface* whose gradient determines the forces governing the dynamic motions of the nuclei. The numerical solution of (8.1) is complicated by the *Pauli principle* which states that the wave function ψ must be antisymmetric in all coordinates representing electrons of equal spin. To state it, we need to clarify that every electron is not only defined by its location but also by its spin which may be positive or negative.

Depending on whether two electrons have the same spin or not, their interaction changes massively. This is reflected by the Pauli principle that we already mentioned: Suppose that electrons i and j have equal spin, then the wave function must satisfy

$$P_{i,j}\psi = -\psi, \tag{8.2}$$

where $P_{i,j}$ denotes the operator that swaps r_i and r_j, i.e., $(P_{i,j}\psi)(r) = \psi(r_1, \dots, r_j, \dots, r_i, \dots, r_n)$. In particular, no two electrons with the same spin can occupy the same location. The challenges associated with solving the Schrödinger equation inspired the following famous quote by Paul Dirac [Dir29]:

> "The fundamental laws necessary for the mathematical treatment of a large part of physics and the whole of chemistry are thus completely known, and the difficulty lies only in the fact that application of these laws leads to equations that are too complex to be solved."

We now describe how deep learning methods might help to mitigate this claim to a certain extent. Let X be a random variable with density $|\psi(r)|^2/\|\psi\|_{L^2}^2$. Using the Rayleigh–Ritz principle, finding the minimal nonzero eigenvalue of (8.1) can be reformulated as minimizing the Rayleigh quotient

$$\frac{\int_{\mathbb{R}^{3\times n}} \overline{\psi(r)}(\mathcal{H}_R\psi)(r)\,\mathrm{d}r}{\|\psi\|_{L^2}^2} = \mathbb{E}\left[\frac{(\mathcal{H}_R\psi)(X)}{\psi(X)}\right] \tag{8.3}$$

over all ψ's satisfying the Pauli principle, see [SO12]. Since this represents a minimization problem it can in principle be solved via a NN ansatz by generating training data distributed according to X using MCMC sampling[31]. Since the wave function ψ will be parametrized as a NN, the minimization of (8.3) will require the computation of the gradient of (8.3) with respect to the NN parameters (the method in [PSMF20] even requires second order derivatives) which, at first sight, might seem to require the computation of third order derivatives. However, due to the Hermitian structure of the Hamiltonian one does not need to compute the derivative of the Laplacian of ψ, see, for example, [HSN20, Equation (8)].

Compared to the other PDE problems we have discussed, an additional complication arises from the need to incorporate structural properties and invariances such as the Pauli principle. Furthermore, empirical evidence shows that it is also necessary to hard code the so-called *cusp conditions* which describe the asymptotic behavior of nearby electrons and electrons close to a nucleus into the NN architecture. A first attempt in this direction has been made in [HZE19] and significantly improved NN architectures have been developed in [HSN20, PSMF20, SRG+21] opening the possibility of accurate ab initio computations for previously intractable molecules. The mathematical properties of this exciting line of work remain largely unexplored. We briefly describe the main ideas behind the NN architecture of [HSN20, SRG+21]. Standard numerical approaches (notably the Multireference Hartree Fock Method, see [SO12]) use a low rank approach to minimize (8.3). Such a low rank approach would approximate ψ by sums of products of *one electron orbitals* $\prod_{i=1}^n \varphi_i(r_i)$ but clearly this does not satisfy the Pauli principle (8.2). In order to ensure the Pauli principle, one constructs so-called *Slater determinants* from one electron orbitals with equal spin. More precisely, suppose that the first n_+ electrons with coordinates $r_1, \dots, r_{n_+}$ have positive spin and the last $n - n_+$ electrons have negative spin. Then any function of the form

$$\det\left((\varphi_i(r_j))_{i,j=1}^{n_+}\right) \cdot \det\left((\varphi_i(r_j))_{i,j=n_++1}^{n}\right) \tag{8.4}$$

satisfies (8.2) and is typically called a Slater determinant. While the Pauli principle establishes an (non-classical) interaction between electrons of equal spin, the so-called *exchange correlation*, electrons with opposite spins are uncorrelated in the representation (8.4). In particular, (8.4) ignores interactions between electrons that arise through Coulomb forces, implying that no nontrivial wavefunction can be accurately represented by a single Slater determinant. To capture physical interactions between different electrons, one needs to use sums of Slater determinants as an ansatz. However, it turns out that the number of such determinants that are needed to guarantee a given accuracy scales very badly with the system size n (to the

[31] Observe that for such sampling methods one can just use the unnormalized density $|\psi(r)|^2$ and thus avoid the computation of the normalization $\|\psi\|_{L^2}^2$.

best of our knowledge the best currently known approximation results are contained in [Yse10], where an n-independent error rate is shown, however the implicit constant in this rate depends at least exponentially on the system size n).

We would like to highlight the approach of [HSN20] whose main idea is to use NNs to incorporate interactions into Slater determinants of the form (8.4) using what is called the *backflow trick* [RMD$^+$06]. The basic building blocks would now consist of functions of the form

$$\det\left((\varphi_i(r_j)\Psi_j(r,\theta_j))_{i,j=1}^{n_+}\right)\cdot\det\left((\varphi_i(r_j)\Psi_j(r,\theta_j))_{i,j=n_++1}^{n}\right), \tag{8.5}$$

where $\Psi_k(\cdot,\theta_k)$, $k\in[n]$, are NNs. If these are arbitrary NNs, it is easy to see that the Pauli principle (8.2) will not be satisfied. However, if we require the NNs to be symmetric, for example, in the sense that for $i,j,s\in[n_+]$ it holds that

$$P_{i,j}\Psi_k(\cdot,\theta_k)=\begin{cases}\Psi_k(\cdot,\theta_k), & \text{if } k\notin\{i,j\},\\ \Psi_i(\cdot,\theta_i), & \text{if } k=j,\\ \Psi_j(\cdot,\theta_j), & \text{if } k=i,\end{cases} \tag{8.6}$$

and analogous conditions hold for $i,j,k\in[n]\setminus[n_+]$, the expression (8.5) does actually satisfy (8.2). The construction of such symmetric NNs can be achieved by using a modification of the so-called *SchNet Architecture* [SKS$^+$17] which can be considered as a specific residual NN.

We describe a simplified construction which is inspired by [HZE19] and used in a slightly more complex form in [SRG$^+$21]. We restrict ourselves to the case of positive spin (e.g., the first n_+ coordinates), the case of negative spin being handled in the same way. Let $\Upsilon(\cdot,\theta^+_{\mathrm{emb}})$ be a univariate NN (with possibly multivariate output) and denote

$$\mathrm{Emb}_k(r,\theta^+_{\mathrm{emb}}):=\sum_{i=1}^{n_+}\Upsilon(\|r_k-r_i\|_2,\theta^+_{\mathrm{emb}}),\quad k\in[n_+],$$

the k-th *embedding layer*. For $k\in[n_+]$, we can now define

$$\Psi_k\left(r,\theta_k\right)=\Psi_k\left(r,(\theta_{k,\mathrm{fc}},\theta^+_{\mathrm{emb}})\right)=\Gamma_k\left(\left(\mathrm{Emb}_k(r,\theta^+_{\mathrm{emb}}),(r_{n_++1},\dots,r_n)\right),\theta_{k,\mathrm{fc}}\right),$$

where $\Gamma_k(\cdot,\theta_{k,\mathrm{fc}})$ denotes a standard FC NN with input dimension equal to the output dimension of Ψ^+ plus the dimension of negative spin electrons. The networks Ψ_k, $k\in[n]\setminus[n_+]$, are defined analogously using different parameters θ^-_{emb} for the embeddings. It is straightforward to check that the NNs Ψ_k, $k\in[n]$, satisfy (8.6) so that the backflow determinants (8.5) satisfy the Pauli principle (8.2).

In [HSN20] the backflow determinants (8.5) are further augmented by a multiplicative correction term, the so-called *Jastrow factor* which is also represented by a specific symmetric NN, as well as a correction term that ensures the validity of the cusp conditions. The results of [HSN20] show that this ansatz (namely using linear combinations of backflow determinants (8.5) instead of plain Slater determinants (8.4)) is vastly more efficient in terms of number of determinants needed to obtain chemical accuracy. The full architecture provides a general purpose NN architecture to represent complicated wave functions. A distinct advantage of this approach is that some parameters (for example, embedding layers) may be shared across different nuclear geometries $R\in\mathbb{R}^{3\times p}$ which allows for the efficient computation of potential energy surfaces [SRG$^+$21], see Figure 8.1. Finally, we would like to highlight the customized NN design that incorporates physical invariances, domain knowledge (for example, in the form of cusp conditions), and existing numerical methods, all of which are required for the method to reach its full potential.

Acknowledgment

The research of JB was supported by the Austrian Science Fund (FWF) under grant I3403-N32. GK acknowledges support from DFG-SPP 1798 Grants KU 1446/21-2 and KU 1446/27-2, DFG-SFB/TR 109 Grant C09, BMBF Grant MaGriDo, and NSF-Simons Foundation Grant SIMONS 81420. The authors would

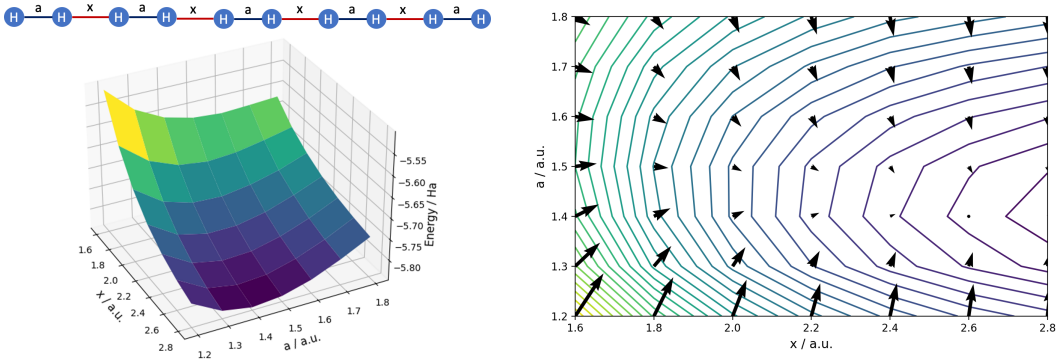


Figure 8.1: By sharing layers across different nuclear geometries one can efficiently compute different geometries in one single training step [SRG+21]. Left: Potential energy surface of H10 chain computed by the deep-learning-based algorithm from [SRG+21]. The lowest energy is achieved when pairs of H atoms enter into a covalent bond to form five H2 molecules. Right: The method of [SRG+21] is capable of accurately computing forces between nuclei which allows for molecular dynamics simulations from first principles.

like to thank Héctor Andrade Loarca, Dennis Elbrächter, Adalbert Fono, Pavol Harar, Lukas Liehr, Duc Anh Nguyen, Mariia Seleznova, and Frieder Simon for their helpful feedback on an early version of this article. In particular, Dennis Elbrächter was providing help for several theoretical results.

References

[AAČ13] Antonio Auffinger, Gérard Ben Arous, and Jiří Černỳ, *Random matrices and complexity of spin glasses*, Communications on Pure and Applied Mathematics **66** (2013), no. 2, 165–201.

[AB99] Martin Anthony and Peter L Bartlett, *Neural network learning: Theoretical foundations*, Cambridge University Press, 1999.

[ACGH19] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu, *A convergence analysis of gradient descent for deep linear neural networks*, International Conference on Learning Representations, 2019.

[ACH18] Sanjeev Arora, Nadav Cohen, and Elad Hazan, *On the optimization of deep networks: Implicit acceleration by overparameterization*, International Conference on Machine Learning, 2018, pp. 372–389.

[AD20] Ben Adcock and Nick Dexter, *The gap between theory and practice in function approximation with deep neural networks*, 2020, arXiv preprint arXiv:2001.07523.

[ADH+19] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang, *On exact computation with an infinitely wide neural net*, Advances in Neural Information Processing Systems, 2019, pp. 8139–8148.

[AGNZ18] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang, *Stronger generalization bounds for deep nets via a compression approach*, International Conference on Machine Learning, 2018, pp. 254–263.

[AHNB+20] Yasmine S Al-Hamdani, Péter R Nagy, Dennis Barton, Mihály Kállay, Jan Gerit Brandenburg, and Alexandre Tkatchenko, *Interactions between large molecules: Puzzle for reference quantum-mechanical methods*, 2020, arXiv preprint arXiv:2009.08927.

[AHS85] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski, *A learning algorithm for Boltzmann machines*, Cognitive Science **9** (1985), no. 1, 147–169.

[AHW96] Peter Auer, Mark Herbster, and Manfred K Warmuth, *Exponentially many local minima for single neurons*, Advances in Neural Information Processing Systems, 1996, p. 316–322.

[AMÖS19] Simon Arridge, Peter Maass, Ozan Öktem, and Carola-Bibiane Schönlieb, *Solving inverse problems using data-driven models*, Acta Numerica **28** (2019), 1–174.

[AÖ17] Jonas Adler and Ozan Öktem, *Solving ill-posed inverse problems using iterative deep neural networks*, Inverse Problems **33** (2017), no. 12, 124007.

[AZLS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song, *A convergence theory for deep learning via over-parameterization*, International Conference on Machine Learning, 2019, pp. 242–252.

[Bar92] Andrew R Barron, *Neural net approximation*, Yale Workshop on Adaptive and Learning Systems, vol. 1, 1992, pp. 69–72.

[Bar93] ———, *Universal approximation bounds for superpositions of a sigmoidal function*, IEEE Transactions on Information Theory **39** (1993), no. 3, 930–945.

[Bar98] Peter L Bartlett, *The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network*, IEEE Transactions on Information Theory **44** (1998), no. 2, 525–536.

[BBC17] Alfred Bourely, John Patrick Boueri, and Krzysztof Choromonski, *Sparse neural networks topologies*, 2017, arXiv preprint arXiv:1706.05683.

[BBC+19] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, and Chris Hesse, *Dota 2 with large scale deep reinforcement learning*, 2019, arXiv preprint arXiv:1912.06680.

[BBG+21] Christian Beck, Sebastian Becker, Philipp Grohs, Nor Jaafari, and Arnulf Jentzen, *Solving the kolmogorov pde by means of deep learning*, Journal of Scientific Computing **88** (2021), no. 3, 1–28.

[BBL03] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi, *Introduction to statistical learning theory*, Summer School on Machine Learning, 2003, pp. 169–207.

[BBL+17] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst, *Geometric deep learning: going beyond euclidean data*, IEEE Signal Processing Magazine **34** (2017), no. 4, 18–42.

[BBM05] Peter L Bartlett, Olivier Bousquet, and Shahar Mendelson, *Local Rademacher complexities*, The Annals of Statistics **33** (2005), no. 4, 1497–1537.

[BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, *Neural machine translation by jointly learning to align and translate*, International Conference on Learning Representations, 2015.

[BDG20] Julius Berner, Markus Dablander, and Philipp Grohs, *Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning*, Advances in Neural Information Processing Systems, 2020, pp. 16615–16627.

[BE02] Olivier Bousquet and André Elisseeff, *Stability and generalization*, Journal of Machine Learning Research **2** (2002), no. Mar, 499–526.

[BEG19] Julius Berner, Dennis Elbrächter, and Philipp Grohs, *How degenerate is the parametrization of neural networks with the ReLU activation function?*, Advances in Neural Information Processing Systems, 2019, pp. 7790–7801.

[Bel52] Richard Bellman, *On the theory of dynamic programming*, Proceedings of the National Academy of Sciences **38** (1952), no. 8, 716.

[BF19] Jan Bohn and Michael Feischl, *Recurrent neural networks as optimal mesh refinement strategies*, 2019, arXiv preprint arXiv:1909.04275.

[BFT17] Peter L Bartlett, Dylan J Foster, and Matus Telgarsky, *Spectrally-normalized margin bounds for neural networks*, Advances in Neural Information Processing Systems, 2017, pp. 6240–6249.

[BGJ20] Julius Berner, Philipp Grohs, and Arnulf Jentzen, *Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of black–scholes partial differential equations*, SIAM Journal on Mathematics of Data Science **2** (2020), no. 3, 631–657.

[BH89] Eric B Baum and David Haussler, *What size net gives valid generalization?*, Neural Computation **1** (1989), no. 1, 151–160.

[BHLM19] Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian, *Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks*, Journal of Machine Learning Research **20** (2019), 63–1.

[BHMM19] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal, *Reconciling modern machine-learning practice and the classical bias–variance trade-off*, Proceedings of the National Academy of Sciences **116** (2019), no. 32, 15849–15854.

[BHX20] Mikhail Belkin, Daniel Hsu, and Ji Xu, *Two models of double descent for weak features*, SIAM Journal on Mathematics of Data Science **2** (2020), no. 4, 1167–1180.

[BK18] Andrew R Barron and Jason M Klusowski, *Approximation and estimation for high-dimensional deep learning networks*, 2018, arXiv preprint arXiv:1809.03090.

[BLLT20] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler, *Benign overfitting in linear regression*, Proceedings of the National Academy of Sciences **117** (2020), no. 48, 30063–30070.

[BMM98] Peter L Bartlett, Vitaly Maiorov, and Ron Meir, *Almost linear VC-dimension bounds for piecewise polynomial networks*, Neural Computation **10** (1998), no. 8, 2159–2173.

[BMM18] Mikhail Belkin, Siyuan Ma, and Soumik Mandal, *To understand deep learning we need to understand kernel learning*, International Conference on Machine Learning, 2018, pp. 541–549.

[BMR+20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei, *Language models are few-shot learners*, Advances in Neural Information Processing Systems, 2020, pp. 1877–1901.

[BR89] Avrim Blum and Ronald L Rivest, *Training a 3-node neural network is NP-complete*, Advances in Neural Information Processing Systems, 1989, pp. 494–501.

[BRT19] Mikhail Belkin, Alexander Rakhlin, and Alexandre B Tsybakov, *Does data interpolation contradict statistical optimality?*, International Conference on Artificial Intelligence and Statistics, 2019, pp. 1611–1619.

[BSW14] Pierre Baldi, Peter Sadowski, and Daniel Whiteson, *Searching for exotic particles in high-energy physics with deep learning*, Nature Communications **5** (2014), no. 1, 1–9.

[BZAJ20] Riccardo Barbano, Chen Zhang, Simon Arridge, and Bangti Jin, *Quantifying model uncertainty in inverse problems via bayesian deep gradient descent*, 2020, arXiv preprint arXiv:2007.09971.

[Can98] Emmanuel J Candès, *Ridgelets: Theory and applications*, Ph.D. thesis, Stanford University, 1998.

[CB20] Lenaic Chizat and Francis Bach, *Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss*, Conference on Learning Theory, 2020, pp. 1305–1338.

[CHM+15] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun, *The loss surfaces of multilayer networks*, International Conference on Artificial Intelligence and Statistics, 2015, pp. 192–204.

[CJLZ19] Minshuo Chen, Haoming Jiang, Wenjing Liao, and Tuo Zhao, *Efficient approximation of deep ReLU networks for functions on low dimensional manifolds*, Advances in Neural Information Processing Systems, 2019, pp. 8174–8184.

[CK20] Alexander Cloninger and Timo Klock, *ReLU nets adapt to intrinsic dimensionality beyond the target domain*, 2020, arXiv preprint arXiv:2008.02545.

[CKP12] Peter G Casazza, Gitta Kutyniok, and Friedrich Philipp, *Introduction to finite frame theory*, Finite Frames: Theory and Applications, Birkhäuser Boston, 2012, pp. 1–53.

[CL19] Wojciech Czaja and Weilin Li, *Analysis of time-frequency scattering transforms*, Applied and Computational Harmonic Analysis **47** (2019), no. 1, 149–171.

[CLA15] Anna Choromanska, Yann LeCun, and Gérard Ben Arous, *Open problem: The landscape of the loss surfaces of multilayer networks*, Conference on Learning Theory, 2015, pp. 1756–1760.

[CLM94] Charles K Chui, Xin Li, and Hrushikesh N Mhaskar, *Neural networks for localized approximation*, Mathematics of Computation **63** (1994), no. 208, 607–623.

[CM18] Charles K Chui and Hrushikesh N Mhaskar, *Deep nets for local manifold learning*, Frontiers in Applied Mathematics and Statistics **4** (2018), 12.

[CMBK20] Lin Chen, Yifei Min, Mikhail Belkin, and Amin Karbasi, *Multiple descent: Design your own generalization curve*, 2020, arXiv preprint arXiv:2008.01036.

[COB19] Lenaic Chizat, Edouard Oyallon, and Francis Bach, *On lazy training in differentiable programming*, Advances in Neural Information Processing Systems, 2019, pp. 2937–2947.

[CPV20] Andrei Caragea, Philipp Petersen, and Felix Voigtlaender, *Neural network approximation and estimation of classifiers with classification boundary in a Barron class*, 2020, arXiv preprint arXiv:2011.09363.

[CS02] Felipe Cucker and Steve Smale, *On the mathematical foundations of learning*, Bulletin of the American Mathematical Society **39** (2002), no. 1, 1–49.

[CvMG+14] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, *Learning phrase representations using rnn encoder–decoder for statistical machine translation*, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1724–1734.

[Cyb89] George Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of Control, Signals and Systems **2** (1989), no. 4, 303–314.

[CZ07] Felipe Cucker and Ding-Xuan Zhou, *Learning theory: an approximation theory viewpoint*, vol. 24, Cambridge University Press, 2007.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, *Imagenet: A large-scale hierarchical image database*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.

[DeV98] Ronald A DeVore, *Nonlinear approximation*, Acta Numerica **7** (1998), 51–150.

[DGL96] Luc Devroye, László Györfi, and Gábor Lugosi, *A probabilistic theory of pattern recognition*, Springer, 1996.

[DHL18] Simon S Du, Wei Hu, and Jason D Lee, *Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced*, Advances in Neural Information Processing Systems, 2018, pp. 384–395.

[DHP20] Ronald DeVore, Boris Hanin, and Guergana Petrova, *Neural network approximation*, 2020, arXiv preprint arXiv:2012.14501.

[Dir29] Paul Adrien Maurice Dirac, *Quantum mechanics of many-electron systems*, Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character **123** (1929), no. 792, 714–733.

[DKMB20] Sören Dittmer, Tobias Kluth, Peter Maass, and Daniel Otero Baguer, *Regularization by architecture: A deep prior approach for inverse problems*, Journal of Mathematical Imaging and Vision **62** (2020), no. 3, 456–470.

[DLL+19] Simon S Du, Jason D Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai, *Gradient descent finds global minima of deep neural networks*, International Conference on Machine Learning, 2019, pp. 1675–1685.

[Don69] William F Donoghue, *Distributions and fourier transforms*, Pure and Applied Mathematics, Academic Press, 1969.

[DPG+14] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio, *Identifying and attacking the saddle point problem in high-dimensional non-convex optimization*, Advances in Neural Information Processing Systems, 2014, pp. 2933–2941.

[DR17] Gintare Karolina Dziugaite and Daniel M Roy, *Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data*, Conference on Uncertainty in Artificial Intelligence, 2017.

[Dre62] Stuart Dreyfus, *The numerical solution of variational problems*, Journal of Mathematical Analysis and Applications **5** (1962), no. 1, 30–45.

[Dud67] Richard M Dudley, *The sizes of compact subsets of hilbert space and continuity of Gaussian processes*, Journal of Functional Analysis **1** (1967), no. 3, 290–330.

[Dud14] ———, *Uniform central limit theorems*, vol. 142, Cambridge University Press, 2014.

[DZPS18] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh, *Gradient descent provably optimizes over-parameterized neural networks*, International Conference on Learning Representations, 2018.

[E17] Weinan E, *A proposal on machine learning via dynamical systems*, Communications in Mathematics and Statistics **5** (2017), no. 1, 1–11.

[EGJS18] Dennis Elbrächter, Philipp Grohs, Arnulf Jentzen, and Christoph Schwab, *DNN expression rate analysis of high-dimensional PDEs: Application to option pricing*, 2018, arXiv preprint arXiv:1809.07669.

[EHJK19] Weinan E, Martin Hutzenthaler, Arnulf Jentzen, and Thomas Kruse, *On multilevel picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations*, Journal of Scientific Computing **79** (2019), no. 3, 1534–1571.

[EHL19] Weinan E, Jiequn Han, and Qianxiao Li, *A mean-field optimal control formulation of deep learning*, Research in the Mathematical Sciences **6** (2019), no. 1, 1–41.

[Elm90] Jeffrey L Elman, *Finding structure in time*, Cognitive Science **14** (1990), no. 2, 179–211.

[EMW19a] Weinan E, Chao Ma, and Lei Wu, *Barron spaces and the compositional function spaces for neural network models*, 2019, arXiv preprint arXiv:1906.08039.

[EMW19b] ———, *A priori estimates of the population risk for two-layer neural networks*, Communications in Mathematical Sciences **17** (2019), no. 5, 1407–1425.

[EMWW20] Weinan E, Chao Ma, Stephan Wojtowytsch, and Lei Wu, *Towards a mathematical understanding of neural network-based machine learning: what we know and what we don't*, 2020, arXiv preprint arXiv:2009.10713.

[EPGB19] Dennis Elbrächter, Dmytro Perekrestenko, Philipp Grohs, and Helmut Bölcskei, *Deep neural network approximation theory*, 2019, arXiv preprint arXiv:1901.02220.

[ES16] Ronen Eldan and Ohad Shamir, *The power of depth for feedforward neural networks*, Conference on Learning Theory, vol. 49, 2016, pp. 907–940.

[EW20a] Weinan E and Stephan Wojtowytsch, *On the Banach spaces associated with multi-layer ReLU networks: Function representation, approximation theory and gradient descent dynamics*, 2020, arXiv preprint arXiv:2007.15623.

[EW20b] ———, *A priori estimates for classification problems using neural networks*, 2020, arXiv preprint arXiv:2009.13500.

[EW20c] ———, *Representation formulas and pointwise properties for Barron functions*, 2020, arXiv preprint arXiv:2006.05982.

[EY18] Weinan E and Bing Yu, *The deep ritz method: a deep learning-based numerical algorithm for solving variational problems*, Communications in Mathematics and Statistics **6** (2018), no. 1, 1–12.

[FB17] Daniel C Freeman and Joan Bruna, *Topology and geometry of half-rectified network optimization*, International Conference on Learning Representations, 2017.

[FC18] Jonathan Frankle and Michael Carbin, *The lottery ticket hypothesis: Finding sparse, trainable neural networks*, International Conference on Learning Representations, 2018.

[FHH+17] Felix A Faber, Luke Hutchison, Bing Huang, Justin Gilmer, Samuel S Schoenholz, George E Dahl, Oriol Vinyals, Steven Kearnes, Patrick F Riley, and O Anatole Von Lilienfeld, *Prediction errors of molecular machine learning models lower than hybrid DFT error*, Journal of Chemical Theory and Computation **13** (2017), no. 11, 5255–5264.

[Fun89] Ken-Ichi Funahashi, *On the approximate realization of continuous mappings by neural networks*, Neural Networks **2** (1989), no. 3, 183–192.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT Press, 2016.

[Gér17] Aurelien Géron, *Hands-on machine learning with scikit-learn and tensorflow: Concepts, tools, and techniques to build intelligent systems*, O'Reilly Media, 2017.

[GH21] Philipp Grohs and Lukas Herrmann, *Deep neural network approximation for high-dimensional parabolic Hamilton-Jacobi-Bellman equations*, 2021, arXiv preprint arXiv:2103.05744.

[GH22] _____, *Deep neural network approximation for high-dimensional elliptic PDEs with boundary conditions*, IMA Journal of Numerical Analysis **42** (2022), no. 3, 2055–2082.

[GHJVW20] Philipp Grohs, Fabian Hornung, Arnulf Jentzen, and Philippe Von Wurstemberger, *A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations*, Memoirs of the American Mathematical Society (2020).

[GHJY15] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan, *Escaping from saddle points—online stochastic gradient for tensor decomposition*, Conference on Learning Theory, 2015, pp. 797–842.

[GJS+20] Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart, *Scaling description of generalization with number of parameters in deep learning*, Journal of Statistical Mechanics: Theory and Experiment (2020), no. 2, 023401.

[GKP20] Ingo Gühring, Gitta Kutyniok, and Philipp Petersen, *Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms*, Analysis and Applications **18** (2020), no. 05, 803–859.

[GKR20] Philipp Grohs, Sarah Koppensteiner, and Martin Rathmair, *Phase retrieval: Uniqueness and stability*, SIAM Review **62** (2020), no. 2, 301–350.

[GL13] Saeed Ghadimi and Guanghui Lan, *Stochastic first-and zeroth-order methods for nonconvex stochastic programming*, SIAM Journal on Optimization **23** (2013), no. 4, 2341–2368.

[GLSS18a] Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nathan Srebro, *Characterizing implicit bias in terms of optimization geometry*, International Conference on Machine Learning, 2018, pp. 1832–1841.

[GLSS18b] _____, *Implicit bias of gradient descent on linear convolutional networks*, Advances in Neural Information Processing Systems, 2018, pp. 9461–9471.

[GMMM21] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari, *Linearized two-layers neural networks in high dimension*, The Annals of Statistics **49** (2021), no. 2, 1029–1054.

[GOW19] Davis Gilton, Greg Ongie, and Rebecca Willett, *Neumann networks for linear inverse problems in imaging*, IEEE Transactions on Computational Imaging **6** (2019), 328–343.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, *Generative adversarial nets*, Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.

[GRK20] Ingo Gühring, Mones Raslan, and Gitta Kutyniok, *Expressivity of deep neural networks*, 2020, arXiv preprint arXiv:2007.04759.

[GRS18] Noah Golowich, Alexander Rakhlin, and Ohad Shamir, *Size-independent sample complexity of neural networks*, Conference On Learning Theory, 2018, pp. 297–299.

[GS20] Lukas Gonon and Christoph Schwab, *Deep ReLU network expression rates for option prices in high-dimensional, exponential Lévy models*, 2020, ETH Zurich SAM Research Report.

[GV21] Philipp Grohs and Felix Voigtlaender, *Proof of the theory-to-practice gap in deep learning via sampling complexity bounds for neural network approximation spaces*, 2021, arXiv preprint arXiv:2104.02746.

[GW08] Andreas Griewank and Andrea Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*, SIAM, 2008.

[GZ84] Evarist Giné and Joel Zinn, *Some limit theorems for empirical processes*, The Annals of Probability (1984), 929–989.

[Han19] Boris Hanin, *Universal function approximation by deep neural nets with bounded width and ReLU activations*, Mathematics **7** (2019), no. 10, 992.

[Hau95] David Haussler, *Sphere packing numbers for subsets of the boolean n-cube with bounded vapnik-chervonenkis dimension*, Journal of Combinatorial Theory, Series A **2** (1995), no. 69, 217–232.

[HH19] Catherine F Higham and Desmond J Higham, *Deep learning: An introduction for applied mathematicians*, SIAM Review **61** (2019), no. 4, 860–891.

[HHJ15] Martin Hairer, Martin Hutzenthaler, and Arnulf Jentzen, *Loss of regularity for Kolmogorov equations*, The Annals of Probability **43** (2015), no. 2, 468–527.

[HJE18] Jiequn Han, Arnulf Jentzen, and Weinan E, *Solving high-dimensional partial differential equations using deep learning*, Proceedings of the National Academy of Sciences **115** (2018), no. 34, 8505–8510.

[HJKN20] Martin Hutzenthaler, Arnulf Jentzen, Thomas Kruse, and Tuan Anh Nguyen, *A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations*, SN Partial Differential Equations and Applications **1** (2020), no. 2, 1–34.

[HLXZ20] Juncai He, Lin Li, Jinchao Xu, and Chunyue Zheng, *ReLU deep neural networks and linear finite elements*, Journal of Computational Mathematics **38** (2020), no. 3, 502–527.

[HMD16] Song Han, Huizi Mao, and William J Dally, *Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding*, International Conference on Learning Representations, 2016.

[HMRT19] Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani, *Surprises in high-dimensional ridgeless least squares interpolation*, 2019, arXiv preprint arXiv:1903.08560.

[Hoe63] Wassily Hoeffding, *Probability inequalities for sums of bounded random variables*, Journal of the American Statistical Association **58** (1963), no. 301, 13–30.

[Hop82] John J Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, Proceedings of the National Academy of Sciences **79** (1982), no. 8, 2554–2558.

[HR19] Boris Hanin and David Rolnick, *Deep ReLU networks have surprisingly few activation patterns*, Advances in Neural Information Processing Systems, 2019, pp. 359–368.

[HRS16] Moritz Hardt, Ben Recht, and Yoram Singer, *Train faster, generalize better: Stability of stochastic gradient descent*, International Conference on Machine Learning, 2016, pp. 1225–1234.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber, *Long short-term memory*, Neural Computation **9** (1997), no. 8, 1735–1780.

[HS17] Boris Hanin and Mark Sellke, *Approximating continuous functions by ReLU nets of minimal width*, 2017, arXiv preprint arXiv:1710.11278.

[HSL+16] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger, *Deep networks with stochastic depth*, European Conference on Computer Vision, 2016, pp. 646–661.

[HSN20] Jan Hermann, Zeno Schätzle, and Frank Noé, *Deep-neural-network solution of the electronic Schrödinger equation*, Nature Chemistry **12** (2020), no. 10, 891–897.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White, *Multilayer feedforward networks are universal approximators*, Neural Networks **2** (1989), no. 5, 359–366.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The elements of statistical learning: Data mining, inference, and prediction*, Springer Series in Statistics, Springer, 2001.

[HV17] Benjamin D Haeffele and René Vidal, *Global optimality in neural network training*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7331–7339.

[HvdG19] Peter Hinz and Sara van de Geer, *A framework for the construction of upper bounds on the number of affine linear regions of ReLU feed-forward neural networks*, IEEE Transactions on Information Theory **65** (2019), 7304–7324.

[HZ94] Geoffrey E Hinton and Richard S Zemel, *Autoencoders, minimum description length, and helmholtz free energy*, Advances in Neural Information Processing Systems **6** (1994), 3–10.

[HZE19] Jiequn Han, Linfeng Zhang, and Weinan E, *Solving many-electron Schrödinger equation using deep neural networks*, Journal of Computational Physics **399** (2019), 108929.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, Proceedings of IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.

[HZRS16] ———, *Deep residual learning for image recognition*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[IS15] Sergey Ioffe and Christian Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, International Conference on Machine Learning, 2015, pp. 448–456.

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler, *Neural tangent kernel: Convergence and generalization in neural networks*, Advances in Neural Information Processing Systems, 2018, pp. 8571–8580.

[JKMB19] Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio, *Predicting the generalization gap in deep networks with margin distributions*, International Conference on Learning Representations, 2019.

[JKNvW20] Arnulf Jentzen, Benno Kuckuck, Ariel Neufeld, and Philippe von Wurstemberger, *Strong error analysis for stochastic gradient descent optimization algorithms*, IMA Journal of Numerical Analysis **41** (2020), no. 1, 455–492.

[JMFU17] Kyong Hwan Jin, Michael T McCann, Emmanuel Froustey, and Michael Unser, *Deep convolutional neural network for inverse problems in imaging*, IEEE Transactions on Image Processing **26** (2017), no. 9, 4509–4522.

[JMS17] Bangti Jin, Peter Maaß, and Otmar Scherzer, *Sparsity regularization in inverse problems*, Inverse Problems **33** (2017), no. 6, 060301.

[JNM+20] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio, *Fantastic generalization measures and where to find them*, International Conference on Learning Representations, 2020.

[Jor90] Michael I Jordan, *Attractor dynamics and parallelism in a connectionist sequential machine*, Artificial neural networks: concept learning, IEEE Press, 1990, pp. 112–127.

[JT19a] Ziwei Ji and Matus Telgarsky, *Gradient descent aligns the layers of deep linear networks*, International Conference on Learning Representations, 2019.

[JT19b] ———, *A refined primal-dual analysis of the implicit bias*, 2019, arXiv preprint arXiv:1906.04540.

[JT20] ———, *Directional convergence and alignment in deep learning*, Advances in Neural Information Processing Systems, 2020, pp. 17176–17186.

[Jud90] Stephen J Judd, *Neural network design and the complexity of learning*, MIT Press, 1990.

[Kel60] Henry J Kelley, *Gradient theory of optimal flight paths*, Ars Journal **30** (1960), no. 10, 947–954.

[KH09] Alex Krizhevsky and Geoffrey Hinton, *Learning multiple layers of features from tiny images*, Tech. report, University of Toronto, 2009.

[KL18] Sham M Kakade and Jason D Lee, *Provably correct automatic subdifferentiation for qualified programs*, Advances in Neural Information Processing Systems, 2018, pp. 7125–7135.

[KL20] Patrick Kidger and Terry Lyons, *Universal approximation with deep narrow networks*, Conference on Learning Theory, 2020, pp. 2306–2327.

[KM97] Marek Karpinski and Angus Macintyre, *Polynomial bounds for VC dimension of sigmoidal and general Pfaffian neural networks*, Journal of Computer and System Sciences **54** (1997), no. 1, 169–176.

[KMN+17] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang, *On large-batch training for deep learning: Generalization gap and sharp minima*, International Conference on Learning Representations, 2017.

[KPRS19] Gitta Kutyniok, Philipp Petersen, Mones Raslan, and Reinhold Schneider, *A theoretical analysis of deep neural networks and parametric PDEs*, 2019, arXiv preprint arXiv:1904.00377.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[KW52] Jack Kiefer and Jacob Wolfowitz, *Stochastic estimation of the maximum of a regression function*, The Annals of Mathematical Statistics **23** (1952), no. 3, 462–466.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE **86** (1998), no. 11, 2278–2324.

[LBD+89] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel, *Backpropagation applied to handwritten zip code recognition*, Neural Computation **1** (1989), no. 4, 541–551.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, *Deep learning*, Nature **521** (2015), no. 7553, 436–444.

[LBN+18] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein, *Deep neural networks as Gaussian processes*, International Conference on Learning Representations, 2018.

[LC19] Guillaume Lample and François Charton, *Deep learning for symbolic mathematics*, International Conference on Learning Representations, 2019.

[LD21] Licong Lin and Edgar Dobriban, *What causes the test error? Going beyond bias-variance via ANOVA*, Journal of Machine Learning Research **22** (2021), no. 155, 1–82.

[LDS89] Yann LeCun, John S Denker, and Sara A Solla, *Optimal brain damage*, Advances in Neural Information Processing Systems, 1989, pp. 598–605.

[Lew43] Kurt Lewin, *Psychology and the process of group living*, The Journal of Social Psychology **17** (1943), no. 1, 113–131.

[Li21] Weilin Li, *Generalization error of minimum weighted norm and kernel interpolation*, SIAM Journal on Mathematics of Data Science **3** (2021), no. 1, 414–438.

[Lin70] Seppo Linnainmaa, *Alogritmin kumulatiivinen pyöristysvirhe yksittäisten pyöristysvirheiden Taylor-kehitelmänä*, Master's thesis, University of Helsinki, 1970.

[LL18] Yuanzhi Li and Yingyu Liang, *Learning overparameterized neural networks via stochastic gradient descent on structured data*, Advances in Neural Information Processing Systems, 2018, pp. 8157–8166.

[LL19] Kaifeng Lyu and Jian Li, *Gradient descent maximizes the margin of homogeneous neural networks*, International Conference on Learning Representations, 2019.

[LLPS93] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken, *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*, Neural Networks **6** (1993), no. 6, 861–867.

[LLS19] Qianxiao Li, Ting Lin, and Zuowei Shen, *Deep learning via dynamical systems: An approximation perspective*, 2019, arXiv preprint arXiv:1912.10382.

[LML+20] Yiping Lu, Chao Ma, Yulong Lu, Jianfeng Lu, and Lexing Ying, *A mean field analysis of deep ResNet and beyond: Towards provably optimization via overparameterization from depth*, International Conference on Machine Learning, 2020, pp. 6426–6436.

[LÖS18] Sebastian Lunz, Ozan Öktem, and Carola-Bibiane Schönlieb, *Adversarial regularizers in inverse problems*, Advances in Neural Information Processing Systems, 2018, pp. 8507–8516.

[LP21] Fabian Laakmann and Philipp Petersen, *Efficient approximation of solutions of parametric linear transport equations by ReLU DNNs*, Advances in Computational Mathematics **47** (2021), no. 1, 1–32.

[LPRS19] Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes, *Fisher–Rao metric, geometry, and complexity of neural networks*, International Conference on Artificial Intelligence and Statistics, 2019, pp. 888–896.

[LR20] Tengyuan Liang and Alexander Rakhlin, *Just interpolate: Kernel "ridgeless" regression can generalize*, The Annals of Statistics **48** (2020), no. 3, 1329–1347.

[LRZ20] Tengyuan Liang, Alexander Rakhlin, and Xiyu Zhai, *On the multiple descent of minimum-norm interpolants and restricted lower isometry of kernels*, Conference on Learning Theory, 2020, pp. 2683–2711.

[LS17] Shiyu Liang and R Srikant, *Why deep neural networks for function approximation?*, International Conference on Learning Representations, 2017.

[LSAH20] Housen Li, Johannes Schwab, Stephan Antholzer, and Markus Haltmeier, *NETT: Solving inverse problems with deep neural networks*, Inverse Problems **36** (2020), no. 6, 065005.

[LSJR16] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht, *Gradient descent only converges to minimizers*, Conference on Learning Theory, 2016, pp. 1246–1257.

[LT91] Michel Ledoux and Michel Talagrand, *Probability in Banach spaces: Isoperimetry and processes*, vol. 23, Springer Science & Business Media, 1991.

[LTY19] Bo Li, Shanshan Tang, and Haijun Yu, *Better approximations of high dimensional smooth functions by deep neural networks with rectified power units*, Communications in Computational Physics **27** (2019), no. 2, 379–411.

[LXS+20] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington, *Wide neural networks of any depth evolve as linear models under gradient descent*, Journal of Statistical Mechanics: Theory and Experiment **2020** (2020), no. 12, 124002.

[Mal12] Stéphane Mallat, *Group invariant scattering*, Communications on Pure and Applied Mathematics **65** (2012), no. 10, 1331–1398.

[Mal16] ———, *Understanding deep convolutional networks*, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **374** (2016), no. 2065, 20150203.

[MAV18] Poorya Mianjy, Raman Arora, and Rene Vidal, *On the implicit bias of dropout*, International Conference on Machine Learning, 2018, pp. 3540–3548.

[McA99] David A McAllester, *Pac-bayesian model averaging*, Conference on Learning Theory, 1999, pp. 164–170.

[McD89] Colin McDiarmid, *On the method of bounded differences*, Surveys in Combinatorics **141** (1989), no. 1, 148–188.

[Men14] Shahar Mendelson, *Learning without concentration*, Conference on Learning Theory, 2014, pp. 25–39.

[Mha96] Hrushikesh N Mhaskar, *Neural networks for optimal approximation of smooth and analytic functions*, Neural Computation **8** (1996), no. 1, 164–177.

[MHR+18] Alexander G de G Matthews, Jiri Hron, Mark Rowland, Richard E Turner, and Zoubin Ghahramani, *Gaussian process behaviour in wide deep neural networks*, International Conference on Learning Representations, 2018.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, *Playing atari with deep reinforcement learning*, 2013, arXiv preprint arXiv:1312.5602.

[MLE21] Vishal Monga, Yuelong Li, and Yonina C Eldar, *Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing*, IEEE Signal Processing Magazine **38** (2021), no. 2, 18–44.

[MM19] Song Mei and Andrea Montanari, *The generalization error of random features regression: Precise asymptotics and double descent curve*, 2019, arXiv preprint arXiv:1908.05355.

[MOPS20] Carlo Marcati, Joost Opschoor, Philipp Petersen, and Christoph Schwab, *Exponential ReLU neural network approximation rates for point and edge singularities*, 2020, ETH Zurich SAM Research Report.

[MP43] Warren S McCulloch and Walter Pitts, *A logical calculus of the ideas immanent in nervous activity*, The Bulletin of Mathematical Biophysics **5** (1943), no. 4, 115–133.

[MP69] Marvin Minsky and Seymour A Papert, *Perceptrons*, MIT Press, 1969.

[MP99] Vitaly Maiorov and Allan Pinkus, *Lower bounds for approximation by MLP neural networks*, Neurocomputing **25** (1999), no. 1-3, 81–91.

[MPCB14] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio, *On the number of linear regions of deep neural networks*, Advances in Neural Information Processing Systems, 2014, pp. 2924–2932.

[MSL+15] Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik, *Deep neural nets as a method for quantitative structure–activity relationships*, Journal of chemical information and modeling **55** (2015), no. 2, 263–274.

[MV03] Shahar Mendelson and Roman Vershynin, *Entropy and the combinatorial dimension*, Inventiones mathematicae **152** (2003), no. 1, 37–55.

[MVSS20] Vidya Muthukumar, Kailas Vodrahalli, Vignesh Subramanian, and Anant Sahai, *Harmless interpolation of noisy data in regression*, IEEE Journal on Selected Areas in Information Theory **1** (2020), no. 1, 67–83.

[MZ20] Andrea Montanari and Yiqiao Zhong, *The interpolation phase transition in neural networks: Memorization and generalization under lazy training*, 2020, arXiv preprint arXiv:2007.12826.

[NBMS17] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro, *Exploring generalization in deep learning*, Advances in Neural Information Processing Systems, 2017, pp. 5947–5956.

[NBS18] Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro, *A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks*, International Conference on Learning Representations, 2018.

[NH17] Quynh Nguyen and Matthias Hein, *The loss surface of deep and wide neural networks*, International Conference on Machine Learning, 2017, pp. 2603–2612.

[NI20] Ryumei Nakada and Masaaki Imaizumi, *Adaptive approximation and generalization of deep neural network with intrinsic dimensionality*, Journal of Machine Learning Research **21** (2020), no. 174, 1–38.

[NJLS09] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro, *Robust stochastic approximation approach to stochastic programming*, SIAM Journal on Optimization **19** (2009), no. 4, 1574–1609.

[NK19] Vaishnavh Nagarajan and J Zico Kolter, *Uniform convergence may be unable to explain generalization in deep learning*, Advances in Neural Information Processing Systems, 2019, pp. 11615–11626.

[NKB+20] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever, *Deep double descent: Where bigger models and more data hurt*, International Conference on Learning Representations, 2020.

[NLG+19] Mor Shpigel Nacson, Jason D Lee, Suriya Gunasekar, Pedro Henrique Pamplona Savarese, Nathan Srebro, and Daniel Soudry, *Convergence of gradient descent on separable data*, International Conference on Artificial Intelligence and Statistics, 2019, pp. 3420–3428.

[NTS14] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro, *In search of the real inductive bias: On the role of implicit regularization in deep learning*, 2014, arXiv preprint arXiv:1412.6614.

[NTS15] ———, *Norm-based capacity control in neural networks*, Conference on Learning Theory, 2015, pp. 1376–1401.

[NW09] Erich Novak and Henryk Woźniakowski, *Approximation of infinitely differentiable multivariate functions is intractable*, Journal of Complexity **25** (2009), no. 4, 398–404.

[NY83] Arkadi Semenovich Nemirovsky and David Borisovich Yudin, *Problem complexity and method efficiency in optimization*, Wiley-Interscience Series in Discrete Mathematics, Wiley, 1983.

[NZGK21] Tenavi Nakamura-Zimmerer, Qi Gong, and Wei Kang, *Adaptive deep learning for high-dimensional Hamilton–Jacobi–Bellman Equations*, SIAM Journal on Scientific Computing **43** (2021), no. 2, A1221–A1247.

[OF96] Bruno A Olshausen and David J Field, *Sparse coding of natural images produces localized, oriented, bandpass receptive fields*, Nature **381** (1996), no. 60, 609.

[OM98] Genevieve B Orr and Klaus-Robert Müller, *Neural networks: tricks of the trade*, Springer, 1998.

[OPS20] Joost Opschoor, Philipp Petersen, and Christoph Schwab, *Deep ReLU networks and high-order finite element methods*, Analysis and Applications (2020), no. 0, 1–56.

[OS19] Kenta Oono and Taiji Suzuki, *Approximation and non-parametric estimation of ResNet-type convolutional neural networks*, International Conference on Machine Learning, 2019, pp. 4922–4931.

[PGZ+18] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean, *Efficient neural architecture search via parameters sharing*, International Conference on Machine Learning, 2018, pp. 4095–4104.

[PKL+17] Tomaso Poggio, Kenji Kawaguchi, Qianli Liao, Brando Miranda, Lorenzo Rosasco, Xavier Boix, Jack Hidary, and Hrushikesh N Mhaskar, *Theory of deep learning III: explaining the non-overfitting puzzle*, 2017, arXiv preprint arXiv:1801.00173.

[PLR+16] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli, *Exponential expressivity in deep neural networks through transient chaos*, Advances in Neural Information Processing Systems, 2016, pp. 3368–3376.

[PMR+17] Tomaso Poggio, Hrushikesh N Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao, *Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review*, International Journal of Automation and Computing **14** (2017), no. 5, 503–519.

[PP92] Etienne Pardoux and Shige Peng, *Backward stochastic differential equations and quasilinear parabolic partial differential equations*, Stochastic partial differential equations and their applications, Springer, 1992, pp. 200–217.

[PRE17] Vardan Papyan, Yaniv Romano, and Michael Elad, *Convolutional neural networks analyzed via convolutional sparse coding*, Journal of Machine Learning Research **18** (2017), no. 1, 2887–2938.

[PRMN04] Tomaso Poggio, Ryan Rifkin, Sayan Mukherjee, and Partha Niyogi, *General conditions for predictivity in learning theory*, Nature **428** (2004), no. 6981, 419–422.

[PRSE18] Vardan Papyan, Yaniv Romano, Jeremias Sulam, and Michael Elad, *Theoretical foundations of deep learning via sparse representations: A multilayer sparse model and its connection to convolutional neural networks*, IEEE Signal Processing Magazine **35** (2018), no. 4, 72–89.

[PRV20] Philipp Petersen, Mones Raslan, and Felix Voigtlaender, *Topological properties of the set of functions generated by neural networks of fixed size*, Foundations of Computational Mathematics (2020), 1–70.

[PSE17] Vardan Papyan, Jeremias Sulam, and Michael Elad, *Working locally thinking globally: Theoretical guarantees for convolutional sparse coding*, IEEE Transactions on Signal Processing **65** (2017), no. 21, 5687–5701.

[PSMF20] David Pfau, James S Spencer, Alexander GDG Matthews, and W Matthew C Foulkes, *Ab initio solution of the many-electron schrödinger equation with deep neural networks*, Physical Review Research **2** (2020), no. 3, 033429.

[PV18] Philipp Petersen and Felix Voigtlaender, *Optimal approximation of piecewise smooth functions using deep ReLU neural networks*, Neural Networks **108** (2018), 296–330.

[PV20] ———, *Equivalence of approximation by convolutional neural networks and fully-connected networks*, Proceedings of the American Mathematical Society **148** (2020), no. 4, 1567–1581.

[REM17] Yaniv Romano, Michael Elad, and Peyman Milanfar, *The little engine that could: Regularization by denoising (red)*, SIAM Journal on Imaging Sciences **10** (2017), no. 4, 1804–1844.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, *U-net: Convolutional networks for biomedical image segmentation*, International Conference on Medical image computing and computer-assisted intervention, 2015, pp. 234–241.

[RH19] Lars Ruthotto and Eldad Haber, *Deep neural networks motivated by partial differential equations*, Journal of Mathematical Imaging and Vision (2019), 1–13.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, *Learning representations by back-propagating errors*, Nature **323** (1986), no. 6088, 533–536.

[RM51] Herbert Robbins and Sutton Monro, *A stochastic approximation method*, The Annals of Mathematical Statistics (1951), 400–407.

[RMD+06] P López Ríos, Ao Ma, Neil D Drummond, Michael D Towler, and Richard J Needs, *Inhomogeneous backflow transformations in quantum Monte Carlo calculations*, Physical Review E **74** (2006), no. 6, 066701.

[Ros58] Frank Rosenblatt, *The perceptron: a probabilistic model for information storage and organization in the brain*, Psychological review **65** (1958), no. 6, 386.

[RPK+17] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein, *On the expressive power of deep neural networks*, International Conference on Machine Learning, 2017, pp. 2847–2854.

[RPK19] Maziar Raissi, Paris Perdikaris, and George E Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics **378** (2019), 686–707.

[RR+07] Ali Rahimi, Benjamin Recht, et al., *Random features for large-scale kernel machines*, Advances in Neural Information Processing Systems, 2007, pp. 1177–1184.

[Rud06] Walter Rudin, *Real and complex analysis*, McGraw-Hill Series in Higher Mathematics, Tata McGraw-Hill, 2006.

[RWK+20] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari, *What's hidden in a randomly weighted neural network?*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020, pp. 11893–11902.

[Sak99] Akito Sakurai, *Tight bounds for the VC-dimension of piecewise polynomial networks*, Advances in Neural Information Processing Systems, 1999, pp. 323–329.

[SCC18] Uri Shaham, Alexander Cloninger, and Ronald R Coifman, *Provable approximation properties for deep neural networks*, Applied and Computational Harmonic Analysis **44** (2018), no. 3, 537–557.

[Sch15] Jürgen Schmidhuber, *Deep learning in neural networks: An overview*, Neural Networks **61** (2015), 85–117.

[SDR14] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński, *Lectures on stochastic programming: modeling and theory*, SIAM, 2014.

[SEJ+20] Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, and Alex Bridgland, *Improved protein structure prediction using potentials from deep learning*, Nature **577** (2020), no. 7792, 706–710.

[SGHK18] David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli, *Analysing mathematical reasoning abilities of neural models*, International Conference on Learning Representations, 2018.

[SGS15] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber, *Training very deep networks*, Advances in Neural Information Processing Systems, 2015, pp. 2377–2385.

[SH19] Johannes Schmidt-Hieber, *Deep ReLU network approximation of functions on a manifold*, 2019, arXiv preprint arXiv:1908.00695.

[She20] Zuowei Shen, *Deep network approximation characterized by number of neurons*, Communications in Computational Physics **28** (2020), no. 5, 1768–1811.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research **15** (2014), no. 1, 1929–1958.

[SHM+16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, and Marc Lanctot, *Mastering the game of go with deep neural networks and tree search*, Nature **529** (2016), no. 7587, 484–489.

[SHN+18] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro, *The implicit bias of gradient descent on separable data*, 2018.

[Ším02] Jiří Šíma, *Training a single sigmoidal neuron is hard*, Neural Computation **14** (2002), no. 11, 2709–2728.

[SKS+17] Kristof T Schütt, Pieter-Jan Kindermans, Huziel E Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller, *Schnet: A continuous-filter convolutional neural network for modeling quantum interactions*, Advances in Neural Information Processing Systems, 2017, pp. 992–1002.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, *Going deeper with convolutions*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[SO12] Attila Szabo and Neil S Ostlund, *Modern quantum chemistry: introduction to advanced electronic structure theory*, Courier Corporation, 2012.

[SPRE18] Jeremias Sulam, Vardan Papyan, Yaniv Romano, and Michael Elad, *Multilayer convolutional sparse modeling: Pursuit and dictionary learning*, IEEE Transactions on Signal Processing **66** (2018), no. 15, 4090–4104.

[SRG+21] Michael Scherbela, Rafael Reisenhofer, Leon Gerard, Philipp Marquetand, and Philipp Grohs, *Solving the electronic Schrödinger equation for multiple nuclear geometries with weight-sharing deep neural networks*, 2021, arXiv preprint arXiv:2105.08351.

[SS16] Itay Safran and Ohad Shamir, *On the quality of the initial basin in overspecified neural networks*, International Conference on Machine Learning, 2016, pp. 774–782.

[SS17] ———, *Depth-width tradeoffs in approximating natural functions with neural networks*, International Conference on Machine Learning, 2017, pp. 2979–2987.

[SS18] ———, *Spurious local minima are common in two-layer ReLU neural networks*, International Conference on Machine Learning, 2018, pp. 4433–4441.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David, *Understanding machine learning: From theory to algorithms*, Cambridge University Press, 2014.

[SSS+17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, and Adrian Bolton, *Mastering the game of go without human knowledge*, Nature **550** (2017), no. 7676, 354–359.

[SSSSS09] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan, *Stochastic convex optimization*, Conference on Learning Theory, 2009.

[STIM18] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry, *How does batch normalization help optimization?*, Advances in Neural Information Processing Systems, 2018, pp. 2488–2498.

[SZ19] Christoph Schwab and Jakob Zech, *Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in uq*, Analysis and Applications **17** (2019), no. 01, 19–55.

[Tal94] Michel Talagrand, *Sharper bounds for Gaussian and empirical processes*, The Annals of Probability (1994), 28–76.

[Tel15] Matus Telgarsky, *Representation benefits of deep feedforward networks*, 2015, arXiv preprint arXiv:1509.08101.

[TvG18] Matthew Thorpe and Yves van Gennip, *Deep limits of residual neural networks*, 2018, arXiv preprint arXiv:1810.11741.

[UVL18] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky, *Deep image prior*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9446–9454.

[Vap99] Vladimir Vapnik, *An overview of statistical learning theory*, IEEE Transactions on Neural Networks **10** (1999), no. 5, 988–999.

[Vap13] ———, *The nature of statistical learning theory*, Springer science & business media, 2013.

[VBB19] Luca Venturi, Afonso S Bandeira, and Joan Bruna, *Spurious valleys in one-hidden-layer neural network optimization landscapes*, Journal of Machine Learning Research **20** (2019), no. 133, 1–34.

[VBC+19] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, and Petko Georgiev, *Grandmaster level in StarCraft II using multi-agent reinforcement learning*, Nature **575** (2019), no. 7782, 350–354.

[VC71] Vladimir Vapnik and Alexey Chervonenkis, *On the uniform convergence of relative frequencies of events to their probabilities*, Theory of Probability & Its Applications **16** (1971), no. 2, 264–280.

[vdVW97] Aad W van der Vaart and Jon A Wellner, *Weak convergence and empirical processes with applications to statistics*, Journal of the Royal Statistical Society-Series A Statistics in Society **160** (1997), no. 3, 596–608.

[Ver18] Roman Vershynin, *High-dimensional probability: An introduction with applications in data science*, vol. 47, Cambridge University Press, 2018.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

[Wer88] Paul J Werbos, *Generalization of backpropagation with application to a recurrent gas market model*, Neural Networks **1** (1988), no. 4, 339–356.

[WGB17] Thomas Wiatowski, Philipp Grohs, and Helmut Bölcskei, *Energy propagation in deep convolutional neural networks*, IEEE Transactions on Information Theory **64** (2017), no. 7, 4819–4842.

[Whi34] Hassler Whitney, *Analytic extensions of differentiable functions defined in closed sets*, Transactions of the American Mathematical Society **36** (1934), no. 1, 63–89.

[WPC+21] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip, *A comprehensive survey on graph neural networks*, IEEE Transactions on Neural Networks and Learning Systems **32** (2021), no. 1, 4–24.

[WZ95] Ronald J Williams and David Zipser, *Gradient-based learning algorithms for recurrent*, Backpropagation: Theory, Architectures, and Applications **433** (1995), 17.

[WZZ+13] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus, *Regularization of neural networks using dropconnect*, International Conference on Machine Learning, 2013, pp. 1058–1066.

[XM12] Huan Xu and Shie Mannor, *Robustness and generalization*, Machine learning **86** (2012), no. 3, 391–423.

[Yan19] Greg Yang, *Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation*, 2019, arXiv preprint arXiv:1902.04760.

[Yar17] Dmitry Yarotsky, *Error bounds for approximations with deep ReLU networks*, Neural Networks **94** (2017), 103–114.

[Yar18a] ———, *Optimal approximation of continuous functions by very deep ReLU networks*, Conference on Learning Theory, 2018, pp. 639–649.

[Yar18b] ———, *Universal approximations of invariant maps by neural networks*, 2018, arXiv preprint arXiv:1804.10306.

[Yar21] ———, *Elementary superexpressive activations*, 2021, arXiv preprint arXiv:2102.10911.

[YGLD17] Rujie Yin, Tingran Gao, Yue M Lu, and Ingrid Daubechies, *A tale of two bases: Local-nonlocal regularization on image patches with convolution framelets*, SIAM Journal on Imaging Sciences **10** (2017), no. 2, 711–750.

[YHC18] Jong Chul Ye, Yoseob Han, and Eunju Cha, *Deep convolutional framelets: A general deep learning framework for inverse problems*, SIAM Journal on Imaging Sciences **11** (2018), no. 2, 991–1048.

[YHPC18] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria, *Recent trends in deep learning based natural language processing*, IEEE Computational Intelligence Magazine **13** (2018), no. 3, 55–75.

[Yse10] Harry Yserentant, *Regularity and approximability of electronic wave functions*, Springer, 2010.

[YZ20] Dmitry Yarotsky and Anton Zhevnerchuk, *The phase diagram of approximation rates for deep neural networks*, Advances in Neural Information Processing Systems, vol. 33, 2020.

[ZAP16] Hao Zhou, Jose M Alvarez, and Fatih Porikli, *Less is more: Towards compact CNNs*, European Conference on Computer Vision, 2016, pp. 662–677.

[Zas75] Thomas Zaslavsky, *Facing up to arrangements: Face-count formulas for partitions of space by hyperplanes: Face-count formulas for partitions of space by hyperplanes*, Memoirs of the American Mathematical Society, American Mathematical Society, 1975.

[ZBH+17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals, *Understanding deep learning requires rethinking generalization*, International Conference on Learning Representations, 2017.

[ZBH+20] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Michael C Mozer, and Yoram Singer, *Identity crisis: Memorization and generalization under extreme overparameterization*, International Conference on Learning Representations, 2020.

[ZBS19] Chiyuan Zhang, Samy Bengio, and Yoram Singer, *Are all layers created equal?*, 2019, arXiv preprint arXiv:1902.01996.

[ZCZG20] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu, *Gradient descent optimizes over-parameterized deep ReLU networks*, Machine Learning **109** (2020), no. 3, 467–492.

[Zho20a] Ding-Xuan Zhou, *Theory of deep convolutional neural networks: Downsampling*, Neural Networks **124** (2020), 319–327.

[Zho20b] ———, *Universality of deep convolutional neural networks*, Applied and Computational Harmonic Analysis **48** (2020), no. 2, 787–794.

[ZKS+18] Jure Zbontar, Florian Knoll, Anuroop Sriram, Tullie Murrell, Zhengnan Huang, Matthew J Muckley, Aaron Defazio, Ruben Stern, Patricia Johnson, Mary Bruno, Marc Parente, Krzysztof J Geras, Joe Katsnelson, Hersh Chandarana, Zizhao Zhang, Michal Drozdzal, Adriana Romero, Michael Rabbat, Pascal Vincent, Nafissa Yakubova, James Pinkerton, Duo Wang, Erich Owens, C Lawrence Zitnick, Michael P Recht, Daniel K Sodickson, and Yvonne W Lui, *fastMRI: An open dataset and benchmarks for accelerated MRI*, 2018, arXiv preprint arXiv:1811.08839.

[ZL17] Barret Zoph and Quoc V Le, *Neural architecture search with reinforcement learning*, International Conference on Learning Representations, 2017.

II Towards a Regularity Theory for ReLU Networks – Chain Rule and Global Error Estimates

Comments

Conference: Oral presentation at SampTA 2019.
E-Print: arXiv:1905.04992[cs.LG]

Contribution

Dennis Elbrächter and Julius Berner contributed equally to the ideas, development, and writing of the paper. The numerical experiments for the figures have been conducted by Julius Berner. Philipp Grohs provided a rough draft for the global estimates and, together with Arnulf Jentzen, assisted with scientific advice.

Bibliographic Information

Berner, Julius, Dennis Elbrächter, Philipp Grohs, and Arnulf Jentzen (2019). "Towards a regularity theory for ReLU networks–chain rule and global error estimates." In: *2019 13th International conference on Sampling Theory and Applications (SampTA)*. IEEE, pp. 1–5. DOI: 10.1109/SampTA45681.2019.9031005.

Copyright Notice

Towards a regularity theory for ReLU networks – chain rule and global error estimates

Julius Berner*, Dennis Elbrächter*, Philipp Grohs‡, Arnulf Jentzen§
*Faculty of Mathematics, University of Vienna
Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
‡Faculty of Mathematics and Research Platform DataScience@UniVienna, University of Vienna
Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
§Department of Mathematics, ETH Zürich
Rämistrasse 101, 8092 Zürich, Switzerland

***Abstract*—Although for neural networks with locally Lipschitz continuous activation functions the classical derivative exists almost everywhere, the standard chain rule is in general not applicable. We will consider a way of introducing a derivative for neural networks that admits a chain rule, which is both rigorous and easy to work with. In addition we will present a method of converting approximation results on bounded domains to global (pointwise) estimates. This can be used to extend known neural network approximation theory to include the study of regularity properties. Of particular interest is the application to neural networks with ReLU activation function, where it contributes to the understanding of the success of deep learning methods for high-dimensional partial differential equations.**

I. Introduction

It has been observed that deep neural networks exhibit the remarkable capability of overcoming the curse of dimensionality in a number of different scenarios. In particular, for certain types of high-dimensional partial differential equations (PDEs) there are promising empirical observations [1], [2], [3], [4], [5], [6], [7] backed by theoretical results for both the approximation error [8], [9], [10], [11] as well as the generalization error [12]. In this context it becomes relevant to not only show how well a given function of interest can be approximated by neural networks but also to extend the study to the derivative of this function. A number of recent publications [13], [14], [15] have investigated the required size of a network which is sufficient to approximate certain interesting (classes of) functions within a given accuracy. This is achieved, first, by considering the approximation of basic functions by very simple networks and, subsequently, by combining those networks in order to approximate more difficult structures. To extend this approach to include the regularity of the approximation, one requires some kind of chain rule for the composition of neural networks. For neural networks with differentiable activation function the standard chain rule is sufficient. It, however, fails when considering neural networks with an activation function, which is not everywhere differentiable. Although locally Lipschitz continuous functions are w.r.t the Lebesgue measure almost everywhere (a.e.) differentiable, the standard chain rule is not applicable, as, in general, it does not hold even in an 'almost everywhere' sense. We will introduce derivatives of neural networks in a way that admits a chain rule which is both rigorous as well as easy to work with. Chain rules for functions which are not everywhere differentiable have been considered in a more general setting in e.g. [16], [17]. We employ the specific structure of neural networks to get stronger results using simpler arguments. In particular it allows for a stability result, i.e. Lemma III.3, the application of which will be discussed in Section V. We would also like to mention a very recent work [18] about approximation in Sobolev norms, where they deal with the issue by using a general bound for the Sobolev norm of the composition of functions from the Sobolev space $W^{1,\infty}$. Note however that this approach leads to a certain factor depending on the dimensions of the domains of the functions, which can be avoided with our method. For ease of exposition, we formulate our results for neural networks with the ReLU activation function. We, however, consider in Section IV how such a chain rule can be obtained for any activation function which is locally Lipschitz continuous (with at most countably many points at which it is not differentiable). In Section V we briefly sketch how the results from Section III can be utilized to get approximation results for certain classes of functions. Subsequently, in Section VI, we present a general method of deriving global error estimates from such approximation results, which are naturally obtained for bounded domains. Ultimately, we discuss how our results can be used to extend known theory, enabling the further study of the approximation of PDE solutions by neural networks.

II. Setting

As in [14], we consider a neural network Φ to be a finite sequence of matrix-vector pairs, i.e.

$$\Phi = ((A_k, b_k))_{k=1}^{L}, \tag{1}$$

where $A_k \in \mathbb{R}^{N_k \times N_{k-1}}$ and $b_k \in \mathbb{R}^{N_k}$ for some depth $L \in \mathbb{N}$ and layer dimensions $N_0, N_1, \ldots, N_L \in \mathbb{N}$. The realization of the neural network Φ is the function $\mathcal{R}\Phi \colon \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ given by

$$\mathcal{R}\Phi = W_L \circ \mathrm{ReLU} \circ W_{L-1} \circ \ldots \circ \mathrm{ReLU} \circ W_1, \tag{2}$$

where $W_k(x) = A_k x + b_k$ for every $x \in \mathbb{R}^{N_k}$ and where

$$\mathrm{ReLU}(x) := (\max\{0, x_1\}, \ldots, \max\{0, x_N\}) \tag{3}$$

for every $x \in \mathbb{R}^N$. We distinguish between a neural network and its realization, since Φ uniquely induces $\mathcal{R}\Phi$, while in general there can be multiple non-trivially different neural networks with the same realization. The representation of a neural network as a structured set of weights as in (1) allows the introduction of notions of network sizes. While there are slight differences between various publications, commonly considered quantities are the depth (i.e. number of affine transformations), the connectivity (i.e. number of non-zero entries of the A_k and b_k), and the weight bound (i.e. maximum of the absolute values of the entries of the A_k and b_k). In [15] it has been shown that these three quantities determine the length of a bit string which is sufficient to encode the network with a prescribed quantization error. In the following let

$$\Phi = ((A_k, b_k))_{k=1}^{L}, \quad \Psi = ((\tilde{A}_k, \tilde{b}_k))_{k=1}^{\tilde{L}} \tag{4}$$

be neural networks with matching dimensions in the sense that $\mathcal{R}\Phi\colon \mathbb{R}^d \to \mathbb{R}^m$ and $\mathcal{R}\Psi\colon \mathbb{R}^m \to \mathbb{R}^n$. We then define their composition as

$$\begin{aligned}\Psi \odot \Phi := \\ \big(((A_k, b_k))_{k=1}^{L-1}, (\tilde{A}_1 A_L, \tilde{A}_1 b_L + \tilde{b}_1), ((\tilde{A}_k, \tilde{b}_k))_{k=2}^{\tilde{L}}\big).\end{aligned} \tag{5}$$

Direct computation shows

$$\mathcal{R}(\Psi \odot \Phi) = \mathcal{R}\Psi \circ \mathcal{R}\Phi. \tag{6}$$

Note that the realization $\mathcal{R}\Phi$ of a neural network Φ is continuous piecewise linear (CPL) as a composition of CPL functions. Consequently, it is Lipschitz continuous and the realization $\mathcal{R}\Phi$ is almost everywhere differentiable by Rademacher's theorem. In particular all three functions in (6) are a.e. differentiable. This, however, is not sufficient to get the derivative of $\mathcal{R}(\Psi \odot \Phi)$ from the derivatives of $\mathcal{R}\Psi$ and $\mathcal{R}\Phi$ by use of the classical chain rule. Consider the very simple counterexample of $u(x) := \mathrm{ReLU}(x)$ and $v(x) := 0$ and formally apply the chain rule, i.e.

$$(D(u \circ v))(x) = (Du)(v(x)) \cdot (Dv)(x). \tag{7}$$

Even though $(Du)(y)$ is well-defined for every $y \in \mathbb{R}\backslash\{0\}$, the expression $(Du)(v(x))$ is defined for no $x \in \mathbb{R}$. In general this problem occurs when the inner function maps a set of positive measure into a set where the derivative of the outer function does not exist. Now in this case, one can directly see that setting $(Du)(0)$ to any arbitrary value would cause (7) to provide the correct result since $(Dv)(x) = 0$.

III. ReLU network derivative

We proceed by defining the derivative of an arbitrary neural network in a way such that it not only coincides a.e. with the derivative of the realization, but also admits a chain rule. To this end let $H\colon \mathbb{R}^N \to \mathbb{R}^{N \times N}$ be the function given by

$$H(x) := \mathrm{diag}(\mathbb{1}_{(0,\infty)}(x_1), \ldots, \mathbb{1}_{(0,\infty)}(x_N)) \tag{8}$$

for every $x = (x_1, \ldots, x_N) \in \mathbb{R}^N$ and let $\mathcal{R}_K\Phi := \mathcal{R}((A_k, b_k))_{k=1}^{K}$. We then define the neural network derivative of Φ as the function $\mathcal{D}\Phi\colon \mathbb{R}^{N_0} \to \mathbb{R}^{N_L \times N_0}$ given by

$$\mathcal{D}\Phi := A_L \cdot H(\mathcal{R}_{L-1}\Phi) \cdot A_{L-1} \cdot \ldots \cdot H(\mathcal{R}_1\Phi) \cdot A_1. \tag{9}$$

Note that this definition is motivated by formally applying the chain rule with the convention that the derivative of $\max\{0, \cdot\}$ is zero at the origin. Now we need to verify that this is justified.

Theorem III.1. *It holds for almost every $x \in \mathbb{R}^d$ that*

$$(\mathcal{D}\Phi)(x) = (D(\mathcal{R}\Phi))(x). \tag{10}$$

Proof. Let $v\colon \mathbb{R}^d \to \mathbb{R}^N$ be a locally Lipschitz continuous function, define $w := \mathrm{ReLU} \circ v$, and

$$L_i := \{x \in \mathbb{R}^d\colon w_i(x) = 0\} = \{x \in \mathbb{R}^d\colon v_i(x) \leq 0\}. \tag{11}$$

We now use an observation about differentiability on level sets (see e.g. [19, Thm 3.3(i)]), which states that

$$(Dw_i)(x) = 0 \quad \text{for almost every } x \in L_i. \tag{12}$$

As $w_i(x) = v_i(x)$ for every $x \in \mathbb{R}^d\backslash L_i$, we get a.e.

$$Dw_i = \mathbb{1}_{\mathbb{R}^d\backslash L_i} \cdot Dv_i = \mathbb{1}_{(0,\infty)}(v_i) \cdot Dv_i \tag{13}$$

and consequently

$$D(\mathrm{ReLU} \circ v) = H(v) \cdot Dv. \tag{14}$$

The claim follows by induction over the layers $K = 1, \ldots, L$ of Φ, using (14) with $v = \mathcal{R}_K\Phi$ for the induction step. □

Note that even for convex $\mathcal{R}\Phi$ the values of $\mathcal{D}\Phi$ on the nullset do not necessarily lie in the respective subdifferentials of $\mathcal{R}\Phi$, as can be seen in Figure 1. Although Theorem III.1 holds regardless of which value is chosen for the derivative of $\max\{0, \cdot\}$ at the origin, no choice will guarantee that all values of $\mathcal{D}\Phi$ lie in the respective subdifferentials of $\mathcal{R}\Phi$. Here we have set the derivative at the origin to zero, following the convention of software implementations for deep learning applications, e.g. TensorFlow and PyTorch. Using (5) and (9) one can verify by direct computation that $\mathcal{D}$ obeys the chain rule.

Corollary III.2. *It holds for every $x \in \mathbb{R}^d$ that*

$$(\mathcal{D}(\Psi \odot \Phi))(x) = (\mathcal{D}\Psi)(\mathcal{R}\Phi(x)) \cdot (\mathcal{D}\Phi)(x). \tag{15}$$

Note that (15) is well-defined as $\mathcal{D}\Psi$ exists everywhere, although it only coincides with $D(\mathcal{R}\Psi)$ almost everywhere. Theorem III.1 however guarantees that we still have a.e.

$$\mathcal{D}(\Psi \odot \Phi) = D(\mathcal{R}(\Psi \odot \Phi)) = D(\mathcal{R}\Psi \circ \mathcal{R}\Phi). \tag{16}$$

Next we provide a technical result dealing with the stability of our chain rule, which will prove to be useful in Section V.

Lemma III.3. *It holds for almost every $x \in \mathbb{R}^d$ that*

$$\lim_{y \to \mathcal{R}\Phi(x)} \big[(\mathcal{D}\Psi)(y) - (\mathcal{D}\Psi)(\mathcal{R}\Phi(x))\big] \cdot (\mathcal{D}\Phi)(x) = 0. \tag{17}$$

Proof. We first show for every locally Lipschitz continuous function $u\colon \mathbb{R}^m \to \mathbb{R}^N$ and for almost every $x \in \mathbb{R}^d$ that

$$\lim_{y \to \mathcal{R}\Phi(x)} [H(u(y)) - H(u(\mathcal{R}\Phi(x)))] \cdot (D(u \circ \mathcal{R}\Phi))(x) = 0. \tag{18}$$

If $u_i(\mathcal{R}\Phi(x)) \neq 0$ we have

$$\lim_{y \to \mathcal{R}\Phi(x)} \mathbb{1}_{(0,\infty)}(u_i(y)) = \mathbb{1}_{(0,\infty)}(u_i(\mathcal{R}\Phi(x))) \tag{19}$$

as u_i is continuous and $\mathbb{1}_{(0,\infty)}$ is continuous on $\mathbb{R}\backslash\{0\}$. Furthermore, [19, Thm 3.3(i)] implies that

$$(D(u_i \circ \mathcal{R}\Phi))(x) = 0 \tag{20}$$

for almost every $x \in \mathbb{R}^d$ with $u_i(\mathcal{R}\Phi(x)) = 0$. Since a finite union of nullsets is again a nullset, this proves the claim (18). The lemma follows by induction over the layers $K = 1, \dots, \tilde{L}$ of Ψ and applying (18) with $u = \mathcal{R}_K\Psi$. □

IV. General Activation Functions

As mentioned in the introduction, it is possible to replace the ReLU activation function in (2) by some locally Lipschitz continuous, component-wise applied function $\varrho\colon \mathbb{R} \to \mathbb{R}$ with an at most countably large set S of points where ϱ is not differentiable. Specifically, one can define the neural network derivative (with activation function ϱ) as in (9) with $\mathbb{1}_{(0,\infty)}(x_i)$ in (8) replaced by

$$(\bar{D}\varrho)(x_i) := \begin{cases} 0, & x_i \in S \\ (D\varrho)(x_i), & \text{else} \end{cases}. \tag{21}$$

The chain rule can, again, be checked by direct computation and it is straightforward to adapt Theorem III.1 to this more general setting by considering the level sets

$$\{x \in \mathbb{R}^d\colon w_i(x) = s\}, \quad s \in S. \tag{22}$$

If additionally $\bar{D}\varrho$ is continuous on $\mathbb{R} \setminus S$, the proof of Lemma III.3 translates without any modifications.

V. Utilization in Approximation Theory

These results can now be employed to bound the L^∞-norm of $\mathcal{D}(\Psi \circ \Phi) - D(u \circ v)$, given corresponding estimates for the approximation of u and v by Ψ and Φ, respectively. Here, one has to take some care when bounding the term

$$\|[\mathcal{D}\Psi \circ \mathcal{R}\Phi - Du \circ \mathcal{R}\Phi]\, \mathcal{D}\Phi\|_{L^\infty} \tag{23}$$

by

$$\|\mathcal{D}\Psi - Du\|_{L^\infty} \|\mathcal{D}\Phi\|_{L^\infty}. \tag{24}$$

Again it can happen that $\mathcal{R}\Phi$ maps a set of positive measure into a nullset where the estimate for the approximation of Du by $\mathcal{D}\Psi$ in the *essential* supremum norm is not valid. However, using the stability result in Lemma III.3 one can for almost every $x \in \mathbb{R}^d$ shift to a sufficiently close point $y \approx \mathcal{R}\Phi(x)$ where the estimate holds. In [13] Yarotsky explicitly constructs networks whose realization is a linear interpolation[1] of the squaring function (see Fig. 1 for illustration), which directly gives an estimate on the approximation rate for the derivatives. These simple networks can then be combined to get networks approximating multiplication, polynomials and eventually, by means of e.g. local Taylor approximation, functions f whose first $n \geq 1$ (weak) derivatives are bounded. This leads to estimates of the form

$$\|f - \mathcal{R}\Phi_{\varepsilon,B}\|_{L^\infty(I_B)} \leq \varepsilon, \tag{25}$$

[1]The interpolation points are uniformly distributed over the domain of approximation and their number grows exponentially with the size of the networks.

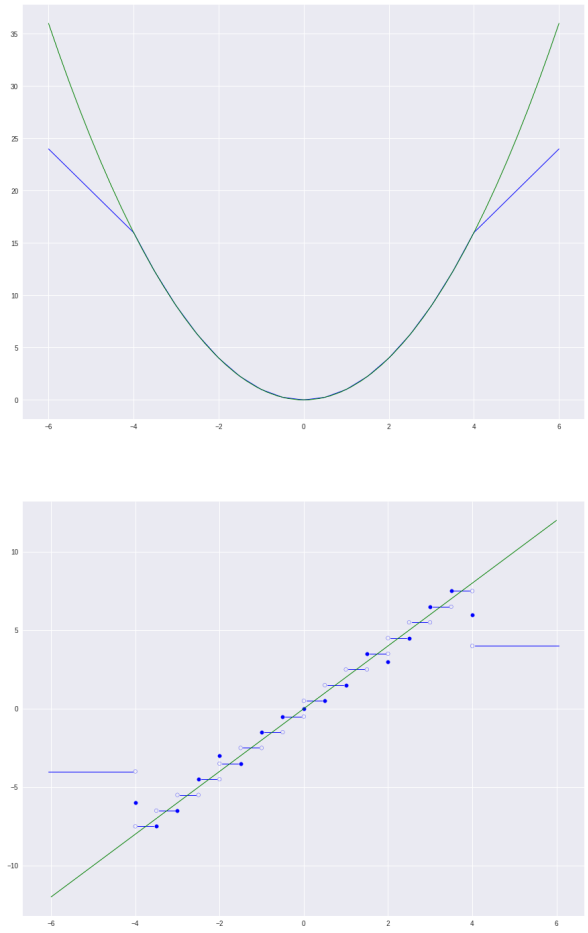


Fig. 1. Approximation of the function $x \mapsto x^2$ and its derivative on the interval $[-4, 4]$ by a neural network Φ with depth 6, connectivity 52 and weight bound 4. Note that not all values of $\mathcal{D}\Phi$ at the points of non-differentiablity of $\mathcal{R}\Phi$ lie between the values at either side, i.e. in the subdifferential.

with $I_B = [-B, B]^d$, including estimates for the scaling of the size of the network $\Phi_{\varepsilon,B}$ w.r.t. B and ε. As these constructions are based on composing simpler functions with known estimates one can now employ Theorem III.1 and Corollary III.2 to show that the derivatives of those networks also approximate the derivative of the function, i.e.

$$\|Df - \mathcal{D}\Phi_{\varepsilon,B}\|_{L^\infty(I_B)} \leq c\,\varepsilon^r. \tag{26}$$

Such constructive approaches can further be found in [8], in [14] for β-cartoon-like functions, in [20] for $(\boldsymbol{b}, \varepsilon)$-holomorphic maps, and in [15] for high-frequent sinusoidal functions.

VI. Global Error Estimates

The error estimates above are usually only sensible for bounded domains, as the realization of a neural network is always CPL with a finite number of pieces. We briefly discuss a general way of transforming them into global pointwise error estimates, which can be useful in the context of PDEs (see e.g.

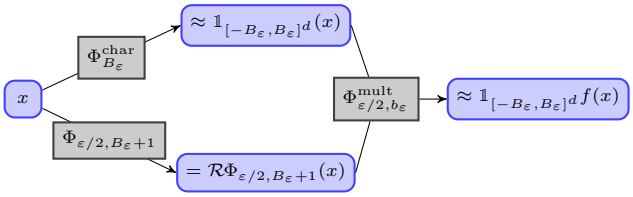


Fig. 2. The neural networks Φ_ε approximating f globally.

[9], [10]). In the following assume that we have a function f with an at most polynomially growing derivative, i.e.

$$\|(Df)(x)\|_2 \leq \boldsymbol{c}(1+\|x\|_2^\kappa). \tag{27}$$

Denote by Φ_B^{char} a neural network which represents the d-dimensional approximate characteristic function of I_B, i.e. $\mathcal{R}\Phi_B^{\mathrm{char}}(x) \in [0,1]$ and

$$\begin{aligned} \mathcal{R}\Phi_B^{\mathrm{char}}(x) &= 1, \quad x \in I_B, \\ \mathcal{R}\Phi_B^{\mathrm{char}}(x) &= 0, \quad x \notin I_{B+1}. \end{aligned} \tag{28}$$

See [15, Proof of Thm. VIII.3] for such a construction. Further let $\Phi^{\mathrm{mult}}_{\varepsilon,b}$ be the neural network approximating the multiplication function on $[-b,b]^2$ with error ε (see e.g. [20, Prop. 3.1]).

Now we define the global approximation networks Φ_ε as the composition of $\Phi^{\mathrm{mult}}_{\varepsilon/2,b_\varepsilon}$ with the parallelization of $\Phi^{\mathrm{char}}_{B_\varepsilon}$ and $\Phi_{\varepsilon/2,B_\varepsilon+1}$ for suitable

$$B_\varepsilon \in \mathcal{O}(\varepsilon^{-1}) \quad \text{and} \quad b_\varepsilon \in \mathcal{O}(\varepsilon^{-\kappa-1}). \tag{29}$$

See Figure 2 for an illustration and e.g. [14, Def. 2.7] for a formal definition of parallelization. Considering the errors on I_B, $I_{B+1}\backslash I_B$ and $\mathbb{R}^d\backslash I_{B+1}$ leads to global estimates, i.e. for every $x \in \mathbb{R}^d$

$$|f(x) - \mathcal{R}\Phi_\varepsilon(x)| \leq \varepsilon(1+\|x\|_2^{\kappa+2}) \tag{30}$$

and, by use of the chain rule III.2, for almost every $x \in \mathbb{R}^d$

$$\|(Df)(x) - (\mathcal{D}\Phi_\varepsilon)(x)\|_2 \leq C\varepsilon^r(1+\|x\|_2^{\kappa+2}). \tag{31}$$

Due to the logarithmic size scaling of the multiplication network, the size of Φ_ε can be bounded by the size of $\Phi_{\varepsilon/2,B_\varepsilon+1}$ plus an additional term in $\mathcal{O}(d+\kappa\log\varepsilon^{-1})$.

VII. Application to PDEs

Analyzing the regularity properties of neural networks was motivated by the recent successful application of deep learning methods to PDEs [2], [3], [4], [5], [6], [7], [11]. Initiated by empirical experiments [1] it has been proven that neural networks are capable of overcoming the curse of dimensionality for solving so-called Kolmogorov PDEs [12]. More precisely, the solution to the empirical risk minimization problem over a class of neural networks approximates the solution of the PDE up to error ε with high probability and with size of the networks and number of samples scaling only polynomially in the dimension d and ε^{-1}. The above requires a suitable learning problem and a sufficiently good approximation of the solution function by neural networks. For Kolmogorov PDEs, this boils down to calculating global Lipschitz coefficients and error estimates for neural networks approximating the initial condition and coefficient functions (see e.g. [9], [10]). Employing estimates of the form (26) one can bound the derivative on I_B, i.e.

$$L_B := \|\mathcal{D}\Phi_{\varepsilon,B}\|_{L^\infty(I_B)} \leq \|Df\|_{L^\infty(I_B)} + c\varepsilon^r. \tag{32}$$

Using mollification and the mean value theorem we can establish local Lipschitz estimates, i.e. for all $x, y \in (-B,B)^d$ that

$$|\mathcal{R}\Phi_{\varepsilon,B}(x) - \mathcal{R}\Phi_{\varepsilon,B}(y)| \leq L_B\|x-y\|_2, \tag{33}$$

and corresponding linear growth bounds

$$|\mathcal{R}\Phi_{\varepsilon,B}(x)| \leq \big(|\mathcal{R}\Phi_{\varepsilon,B}(0)| + L_B\big)(1+\|x\|_2). \tag{34}$$

Similarly, one can use (31) to obtain estimates of the form

$$|\mathcal{R}\Phi_\varepsilon(x) - \mathcal{R}\Phi_\varepsilon(y)| \leq \boldsymbol{C}(1+\|x\|_2^{\kappa+2}+\|y\|_2^{\kappa+2})\|x-y\|_2 \tag{35}$$

for all $x, y \in \mathbb{R}^d$ (which are demanded in [10, Theorem 1.1]). Moreover, note that the capability to produce approximation results which include error estimates for the derivative is of significant independent interest. Various numerical methods (for instance Galerkin methods) rely on bounding the error in some Sobolev norm $\|\cdot\|_{W^{1,p}}$, which requires estimates of the derivative differences. We believe that the possibility to obtain regularity estimates significantly contributes to the mathematical theory of neural networks and allows for further advances in the numerical approximation of high dimensional partial differential equations.

VIII. Relation to backpropagation in training

The approach discussed here could further be applied to the training of neural networks by (stochastic) gradient descent. Note, however, that this is a slightly different setting. From the approximation theory perspective we were interested in the derivative of $x \mapsto \mathcal{R}\Phi(x)$, while in training one requires the derivative of $\Phi \mapsto \mathcal{R}\Phi(x^*)$ for some fixed sample x^*. In particular this function is no longer CPL but rather continuous piecewise polynomial. While this would necessitate some technical modifications, we believe that it should be possible to employ the method used here in order to show that the gradient of $\Phi \mapsto \mathcal{R}\Phi(x^*)$ coincides a.e. with what is computed by backpropagation using the convention of setting the derivative of $\max\{0,\cdot\}$ to zero at the origin (as well as similar conventions for e.g. max-pooling).

Acknowledgment

The research of JB and DE was supported by the Austrian Science Fund (FWF) under grants I3403-N32 and P 30148.

References

[1] C. Beck, S. Becker, P. Grohs, N. Jaafari, and A. Jentzen, "Solving stochastic differential equations and Kolmogorov equations by means of deep learning," *arXiv:1806.00421*, 2018.

[2] W. E, J. Han, and A. Jentzen, "Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations," *Communications in Mathematics and Statistics*, vol. 5, no. 4, pp. 349–380, 2017.

[3] J. Han, A. Jentzen, and W. E, "Solving high-dimensional partial differential equations using deep learning," *arXiv:1707.02568*, 2017.

[4] J. Sirignano and K. Spiliopoulos, "DGM: A deep learning algorithm for solving partial differential equations," *arXiv:1708.07469*, 2017.

[5] M. Fujii, A. Takahashi, and M. Takahashi, "Asymptotic Expansion as Prior Knowledge in Deep Learning Method for high dimensional BSDEs," *arXiv:1710.07030*, 2017.

[6] Y. Khoo, J. Lu, and L. Ying, "Solving parametric PDE problems with artificial neural networks," *arXiv:1707.03351*, 2017.

[7] W. E and B. Yu, "The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems," *arXiv:1710.00211*, 2017.

[8] D. Elbrächter, P. Grohs, A. Jentzen, and C. Schwab, "DNN Expression Rate Analysis of high-dimensional PDEs: Application to Option Pricing," *arXiv:1809.07669*, 2018.

[9] P. Grohs, F. Hornung, A. Jentzen, and P. von Wurstemberger, "A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations," *arXiv:1809.02362*, 2018.

[10] A. Jentzen, D. Salimova, and T. Welti, "A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients," *arxiv:1809.07321*, 2018.

[11] M. Hutzenthaler, A. Jentzen, T. Kruse, and T. A. Nguyen, "A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations," *arXiv:1707.02568*, 2019.

[12] J. Berner, P. Grohs, and A. Jentzen, "Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations," *arXiv:1809.03062*, 2018.

[13] D. Yarotsky, "Optimal approximation of continuous functions by very deep ReLU networks," *arXiv:1802.03620*, 2018.

[14] P. Petersen and F. Voigtlaender, "Optimal approximation of piecewise smooth functions using deep ReLU neural networks," *arXiv:1709.05289*, 2017.

[15] P. Grohs, D. Perekrestenko, D. Elbrächter, and H. Bölcskei, "Deep Neural Network Approximation Theory," *arxiv:1901.02220*, 2019.

[16] F. Murat and C. Trombetti, "A chain rule formula for the composition of a vector-valued function by a piecewise smooth function," *Bollettino dell'Unione Matematica Italiana*, vol. 6, no. 3, pp. 581–595, 2003.

[17] L. Ambrosio and G. Dal Maso, "A general chain rule for distributional derivatives," *Proceedings of the American Mathematical Society*, vol. 108, no. 3, pp. 691–702, 1990.

[18] I. Gühring, G. Kutyniok, and P. Petersen, "Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms," *arXiv:1902.07896*, 2019.

[19] L. C. Evans and R. F. Gariepy, *Measure Theory and Fine Properties of Functions, Revised Edition*, ser. Textbooks in Mathematics. CRC Press, 2015.

[20] C. Schwab and J. Zech, "Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ," *Analysis and Applications, Singapore*, vol. 17, no. 1, pp. 19–55, 2019.

III Analysis of the Generalization Error: Empirical Risk Minimization over Deep Artificial Neural Networks Overcomes the Curse of Dimensionality in the Numerical Approximation of Black–Scholes Partial Differential Equations

Comments

Conference: Oral presentation at GAMM 2019.
E-Print: arXiv:1809.03062[cs.LG]

Contribution

Developed and written by Julius Berner, partly based on a rough outline by Philipp Grohs. Motivation and scientific advice has been provided by Arnulf Jentzen.

Bibliographic Information

Berner, Julius, Philipp Grohs, and Arnulf Jentzen (2020). "Analysis of the Generalization Error: Empirical Risk Minimization over Deep Artificial Neural Networks Overcomes the Curse of Dimensionality in the Numerical Approximation of Black–Scholes Partial Differential Equations." In: *SIAM Journal on Mathematics of Data Science* 2.3, pp. 631–657. DOI: 10.1109/IWOBI.2017.7985525.

Copyright Notice

Analysis of the Generalization Error: Empirical Risk Minimization over Deep Artificial Neural Networks Overcomes the Curse of Dimensionality in the Numerical Approximation of Black–Scholes Partial Differential Equations

Julius Berner*, Philipp Grohs†, and Arnulf Jentzen‡

Abstract. The development of new classification and regression algorithms based on empirical risk minimization (ERM) over deep neural network hypothesis classes, coined deep learning, revolutionized the area of artificial intelligence, machine learning, and data analysis. In particular, these methods have been applied to the numerical solution of high-dimensional partial differential equations with great success. Recent simulations indicate that deep learning–based algorithms are capable of overcoming the curse of dimensionality for the numerical solution of Kolmogorov equations, which are widely used in models from engineering, finance, and the natural sciences. The present paper considers under which conditions ERM over a deep neural network hypothesis class approximates the solution of a d-dimensional Kolmogorov equation with affine drift and diffusion coefficients and typical initial values arising from problems in computational finance up to error ε. We establish that, with high probability over draws of training samples, such an approximation can be achieved with both the size of the hypothesis class and the number of training samples scaling only polynomially in d and ε^{-1}. It can be concluded that ERM over deep neural network hypothesis classes overcomes the curse of dimensionality for the numerical solution of linear Kolmogorov equations with affine coefficients.

Key words. deep learning, curse of dimensionality, Kolmogorov equation, generalization error, empirical risk minimization

AMS subject classifications. 60H30, 65C30, 62M45, 68T05

1. Introduction.

In this introductory section we want to present and motivate our problem, provide the reader with background knowledge and references to previous research on the topic, and outline the important steps, as well as possible extensions, of our contribution.

1.1. Problem statement.

Suppose we need to numerically approximate the end value[1] $F_d(T,\cdot)$ at time $T \in (0,\infty)$ of the solution $F_d \in \mathcal{C}([0,T]\times\mathbb{R}^d,\mathbb{R})$ of a *linear Kolmogorov equation* which for an initial value $\varphi_d \in \mathcal{C}(\mathbb{R}^d,\mathbb{R})$, drift coefficient $\mu_d \in \mathcal{C}(\mathbb{R}^d,\mathbb{R}^d)$, and diffusion coefficient $\sigma_d \in \mathcal{C}(\mathbb{R}^d,\mathbb{R}^{d\times d})$ is defined as[2]

$$\begin{cases} \frac{\partial F_d}{\partial t}(t,x) = \frac{1}{2}\mathrm{Trace}\big(\sigma_d(x)[\sigma_d(x)]^*(\mathrm{Hess}_x F_d)(t,x)\big) + \mu_d(x)\cdot(\nabla_x F_d)(t,x) \\ F_d(0,x) = \varphi_d(x) \end{cases} \tag{1.1}$$

*Faculty of Mathematics, University of Vienna, Austria (julius.berner@univie.ac.at).

†Faculty of Mathematics and Research Platform DataScience@UniVienna, University of Vienna, Austria (philipp.grohs@univie.ac.at).

‡Department of Mathematics, ETH Zürich, Switzerland, and Faculty of Mathematics and Computer Science, University of Münster, Germany (ajentzen@uni-muenster.de).

Funding: This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC 2044-390685587, Mathematics Münster: Dynamics-Geometry-Structure and by the Austrian Science Fund (FWF) under grant I3403-N32.

[1] We write the subscript d as we are interested in approximation rates w.r.t. to the dimension $d \in \mathbb{N}$.

[2] For $x \in \mathbb{R}^d$, $y \in \mathbb{R}^d$ we denote by $x \cdot y := \sum_{i=1}^d x_i y_i$ the standard scalar product of x and y.

for every $(t,x) \in [0,T] \times \mathbb{R}^d$. Important special cases include the heat equation or the Black–Scholes equation from computational finance. For the latter partial differential equation (PDE) typically the coefficients σ_d and μ_d are affine and the initial value φ_d can be represented as a composition of minima, maxima, and linear combinations such as

$$\varphi_d(x) = \min\big\{ \max\{D - c_d \cdot x, 0\}, D\big\} \tag{1.2}$$

with suitable coefficients $D \in (0,\infty)$, $c_d \in \mathbb{R}^d$ in the case of a European put option pricing problem. It is well known that standard numerical methods for solving PDEs, in particular those based on a discretization of the domain, suffer from the curse of dimensionality, meaning that their computational complexity grows exponentially in the dimension d [3, 57].

If the goal is simply to evaluate $F_d(T,\cdot)$ *at a single value* $\xi \in \mathbb{R}^d$, then under suitable assumptions Monte Carlo sampling methods are capable of overcoming the curse of dimensionality. These methods are based on the integral representation (Feynman–Kac formula)

$$F_d(T,\xi) = \mathbb{E}\big[\varphi_d(S_T^\xi)\big], \tag{1.3}$$

where $(S_t^\xi)_{t\in[0,T]}$ is a stochastic process satisfying the stochastic differential equation (SDE)

$$dS_t^\xi = \sigma_d(S_t^\xi)dB_t^d + \mu_d(S_t^\xi)dt \quad \text{and} \quad S_0^\xi = \xi$$

for a d-dimensional $(\mathcal{G}_t)$-Brownian motion B^d on some filtered probability space $(\Omega, \mathcal{G}, \mathbb{P}, (\mathcal{G}_t))$. The evaluation of $F_d(T,\xi)$ can then be computed by approximating the expectation in (1.3) by Monte Carlo integration, that is, by simulating i.i.d. samples $(S^{(i)})_{i=1}^m$ drawn from the distribution of S_T^ξ and by approximating $F_d(T,\xi)$ with the empirical average $\frac{1}{m}\sum_{i=1}^m \varphi_d(S^{(i)})$. It is a standard result that the number of samples m needed to obtain a desired accuracy ε depends only polynomially on the dimension d and ε^{-1} [28].

If the goal, however, is to approximate $F_d(T,\cdot)$ not only at a single value but, for example, on a full hypercube $[u,v]^d$, there has been no known method which does not suffer from the curse of dimensionality. In particular, it has been completely out of range to provably approximate $F_d(T,\cdot)$ on $[u,v]^d$ in high dimensions, say, $d \gg 100$.

The present paper introduces and analyzes deep learning–based algorithms for the numerical approximation of $F_d(T,\cdot)$ on a full hypercube $[u,v]^d$. We will prove that the resulting algorithms overcome the curse of dimensionality and can consequently be efficiently applied even in high dimensions. Our proofs will be based on tools from statistical learning theory and the following key properties of linear Kolmogorov equations:

- P.1 The fact that one can reformulate (1.1) as a learning problem (see Lemma 3.2).
- P.2 The fact that typical initial values arising from problems in computational finance, such as, for example, (1.2), are either exactly representable as neural networks with ReLU activation function (ReLU networks) or can be approximated by such neural networks without incurring the curse of dimensionality (see [29, Section 4]).
- P.3 The fact that Property P.2 is preserved under the evolution of linear Kolmogorov equations (1.1) with affine diffusion and drift coefficients, which implies that $F_d(T,\cdot)$ can be approximated by ReLU networks without incurring the curse of dimensionality (see Theorem 3.3).

1.2. Deep learning and statistical learning theory. In their most basic incarnation, deep learning–based algorithms start with training data

$$((X_d^{(i)}, Y_d^{(i)}))_{i=1}^m \colon \Omega \to ([u,v]^d \times [-D,D])^m.$$

To give a concrete example, $X_d^{(i)}$ may consist of different 28×28 pixel grayscale images of handwritten digits and $Y_d^{(i)}$ may consist of corresponding probabilities describing the likelihood of a certain digit to be shown in image $X_d^{(i)}$ [44]. The goal is then to find a functional relation between images and labels and use it for predictive purposes on unseen images.

Empirical risk minimization (ERM) attempts to solve this prediction problem by minimizing the empirical risk

$$f \mapsto \widehat{\mathcal{E}}_{d,m}(f) := \tfrac{1}{m} \sum_{i=1}^m \big(f(X_d^{(i)}) - Y_d^{(i)}\big)^2 \tag{1.4}$$

over a compact[3] hypothesis class $\mathcal{H} \subseteq \mathcal{C}([u,v]^d, \mathbb{R})$, resulting in a predictor

$$\widehat{f}_{d,m,\mathcal{H}} \in \operatorname*{argmin}_{f \in \mathcal{H}} \widehat{\mathcal{E}}_{d,m}(f)$$

that is hoped to provide a good approximation of the desired functional relation in the training data. In deep learning, these hypothesis classes consist of deep neural networks with fixed activation function $\rho \in \mathcal{C}(\mathbb{R}, \mathbb{R})$, parameter bound $R \in (0, \infty)$, and architecture[4] $\mathbf{a} = (a_0, a_1, \ldots, a_L) \in \mathbb{N}^{L+1}$, where $a_0 = d$ and $a_L = 1$. We define the corresponding set of neural network parametrizations

$$\mathcal{P}_{\mathbf{a},R} := \bigtimes_{l=1}^L \big([-R,R]^{a_l \times a_{l-1}} \times [-R,R]^{a_l}\big),$$

and for a parametrization $\boldsymbol{\theta} = ((W_l, B_l))_{l=1}^L \in \mathcal{P}_{\mathbf{a},R}$ we define its realization function[5]

$$\mathcal{F}_\rho(\boldsymbol{\theta}) := \mathcal{A}_{W_L,B_L} \circ \rho_* \circ \mathcal{A}_{W_{L-1},B_{L-1}} \circ \rho_* \circ \cdots \circ \rho_* \circ \mathcal{A}_{W_1,B_1} \in \mathcal{C}(\mathbb{R}^d, \mathbb{R}),$$

where $\mathcal{A}_{W,B}(x) := Wx + B$ and $\rho_*(x) = (\rho(x_i))_{i=1}^n$, i.e., ρ is applied componentwise. Then, neural network hypothesis classes are typically of the form

$$\mathcal{N}_{\rho,\mathbf{a},R}^{u,v} := \Big\{ \Big([u,v]^d \ni x \mapsto \mathcal{F}_\rho(\boldsymbol{\theta})(x) \Big) : \boldsymbol{\theta} \in \mathcal{P}_{\mathbf{a},R} \Big\}. \tag{1.5}$$

[3]Note that we equip $\mathcal{C}([u,v]^d, \mathbb{R})$ with the uniform norm $\|\cdot\|_{\mathcal{L}^\infty}$ which for $f \in \mathcal{C}([u,v]^d, \mathbb{R})$ is given by $\|f\|_{\mathcal{L}^\infty} = \|f\|_{\mathcal{L}^\infty([u,v]^d)} := \max_{x \in [u,v]^d} |f(x)|$.

[4]Typically one calls a neural network "deep" if the architecture satisfies $L > 2$.

[5]If there is no possibility of ambiguity, we use the term "neural network" interchangeably for the parametrization and the realization function. However, note that $\boldsymbol{\theta} \in \mathcal{P}_{\mathbf{a},R}$ uniquely induces $\mathcal{F}_\rho(\boldsymbol{\theta}) \in \mathcal{C}(\mathbb{R}^d, \mathbb{R})$, while in general there can be multiple nontrivially different parametrizations with the same realization function; see [13].

Despite the great practical success of the "deep learning paradigm," for generic real world training data it is far out of reach to specify network architectures which guarantee a desired performance on unseen data; see also [63].

This type of problem can be theoretically studied using tools developed within the field of statistical learning theory. There it is typically postulated that $((X_d^{(i)}, Y_d^{(i)}))_{i=1}^m$ are i.i.d. samples drawn from the distribution of some (unknown) data (X_d, Y_d) and that the optimal functional relation between X_d and Y_d is given by the regression function

$$f_d^* : \begin{cases} [u,v]^d & \to & \mathbb{R} \\ x & \mapsto & \mathbb{E}\left[Y_d \middle| X_d = x\right], \end{cases}$$

which minimizes the risk $f \mapsto \mathcal{E}_d(f) := \mathbb{E}\left[\left(f(X_d) - Y_d\right)^2\right]$ (see Lemma 2.2). The minimization of functionals of the form $\mathcal{E}_d$ is commonly referred to as a

statistical learning problem with data (X_d, Y_d) and quadratic loss function.

Under strong regularity assumptions on the regression function f_d^* and the distribution of (X_d, Y_d) it is possible to obtain bounds on the sample size m and the "complexity" of the hypothesis class $\mathcal{H}$ in order to guarantee, with high probability, an error

$$\mathbb{E}\left[\left(\widehat{f}_{d,m,\mathcal{H}}(X_d) - f_d^*(X_d)\right)^2\right] \leq \varepsilon; \tag{1.6}$$

see, for example, [4, 7, 18, 19, 31, 41, 47, 61]. In the above, the regularity of the regression function f_d^* quantifies how well f_d^* can be approximated by the hypothesis class $\mathcal{H}$.

In the case of the neural network hypothesis classes $\mathcal{H} = \mathcal{N}_{\rho,\mathbf{a},R}^{u,v}$ these regularity assumptions are met if f_d^* satisfies certain smoothness assumptions; see [15, 16, 26, 52, 54, 58, 62]. Moreover, the complexity of $\mathcal{N}_{\rho,\mathbf{a},R}^{u,v}$ can mainly be described by the size of the neural network parametrizations, i.e., the number of network parameters

$$P(\mathbf{a}) := \textstyle\sum_{l=1}^{L} a_l a_{l-1} + a_l.$$

We will show in Subsection 1.3 below that our specific deep learning–based method for numerically solving Kolmogorov equations allows us to rigorously apply tools from statistical learning theory, as we can overcome the following potential problems:

R.1 The crucial assumption that the training data consists of i.i.d. samples drawn from an underlying probability distribution is usually debatable or at least hard to verify.

R.2 Even if this assumption were satisfied, the underlying distribution of (X_d, Y_d) is typically unknown. Thus, it is hard to ensure a priori the regularity assumptions on f_d^*, which are needed to apply tools from statistical learning theory.

R.3 Since the distribution of X_d is typically unknown, it is not clear how the quantity $\mathbb{E}\left[\left(\widehat{f}_{d,m,\mathcal{H}}(X_d) - f_d^*(X_d)\right)^2\right]$ of (1.6) can be interpreted.

R.4 Most of the classical techniques operate in an asymptotic regime where the number of training samples m exceeds the network size $P(\mathbf{a})$. However, in many applications the number of training samples is fixed, and it is not possible to generate more training data at will.

1.3. Kolmogorov equations as learning problem. We will reformulate the numerical approximation of $F_d(T,\cdot)$ on $[u,v]^d$ as a statistical learning problem and demonstrate that in this specific case none of the aforementioned Problems R.1–R.4 appears. Let $X_d \sim \mathcal{U}([u,v]^d)$ be uniformly distributed on $[u,v]^d$ and define $Y_d := \varphi_d(S_T^{X_d})$, where $(S_t^{X_d})_{t\in[0,T]}$ is a stochastic process satisfying the SDE

$$dS_t^{X_d} = \sigma_d(S_t^{X_d})dB_t^d + \mu_d(S_t^{X_d})dt \quad \text{and} \quad S_0^{X_d} = X_d. \tag{1.7}$$

Under suitable conditions it then follows from the Feynman–Kac formula (1.3) that $F_d(T,\cdot)$ is the minimizer of the risk functional $\mathcal{E}_d$, that is, $f_d^*(x) = F_d(T,x)$ for a.e. $x \in [u,v]^d$; see Lemma 3.2. As outlined in Subsection 1.2, we thus have that

> *the end value of the Kolmogorov equation $F_d(T,\cdot)$ is the solution to the statistical learning problem with data $(X_d, \varphi_d(S_T^{X_d}))$ and quadratic loss function.*

A natural next step is to apply the deep learning paradigm, that is, for m i.i.d. samples $((X_d^{(i)}, Y_d^{(i)}))_{i=1}^m$ drawn from the distribution of (X_d, Y_d) to minimize the empirical risk (1.4) over a hypothesis class of neural networks $\mathcal{N}_{\rho,\mathbf{a},R}^{u,v}$.

In [10] this idea has been implemented with suitable classes of deep neural networks of a given architecture as hypothesis class $\mathcal{H}$. In extensive numerical simulations it was observed that the proposed algorithm is efficient even in very high dimensions, suggesting that it does not suffer from the curse of dimensionality. Similar conclusions can be found in related work [11, 12, 21, 22, 25, 33, 34, 60], which covers topics ranging from American option pricing problems to fully nonlinear PDEs. Note that in [60] a nonquantitative analysis of the approximation error is given; all the other works are purely empirical. To the best of our knowledge, this is the first joint quantitative analysis of approximation and generalization error confirming the efficiency of deep learning–based methods applied to the numerical solution of high-dimensional PDEs.

Two main parameters influence the complexity of the algorithm described above: the number $P(\mathbf{a}_{d,\varepsilon})$ of network parameters that need to be optimized as well as the number of training samples $m_{d,\varepsilon}$ needed to guarantee that, with high probability, the estimate

$$\tfrac{1}{(v-u)^d}\big\|\widehat{f}_{d,m_{d,\varepsilon},\mathcal{H}_{d,\varepsilon}} - F_d(T,\cdot)\big\|^2_{\mathcal{L}^2([u,v]^d)} = \mathbb{E}\left[\big(\widehat{f}_{d,m_{d,\varepsilon},\mathcal{H}_{d,\varepsilon}}(X_d) - F_d(T,X_d)\big)^2\right] \le \varepsilon \tag{1.8}$$

holds true. We are interested in the scaling of $m_{d,\varepsilon}$ and $P(\mathbf{a}_{d,\varepsilon})$ with respect to the precision ε and dimension d.

Observe that the data distribution $(X_d, \varphi_d(S_T^{X_d}))$ is now explicitly known and i.i.d. samples of this distribution can be simulated as needed (X_d is uniformly distributed and can be simulated using a suitable random number generator, and $S_T^{X_d}$ can be simulated by any numerical solver for the SDE (1.7); see [28]). Moreover, the uniform distribution of the input data X_d gives rise to typical $\mathcal{L}^2$-error estimates (1.8) and, in the case of affine coefficients and suitable initial values, we can establish bounds on how well the regression function can be approximated by neural network hypothesis classes; see Theorem 3.3. In particular, contrary to conventional learning problems, in the statistical learning problem that arises from our reformulation of the Kolmogorov equation, none of the Problems R.1–R.4 described in Subsection 1.2 occurs. We will therefore be able to rigorously invoke tools from statistical learning theory to obtain bounds on the quantities $m_{d,\varepsilon}$ and $P(\mathbf{a}_{d,\varepsilon})$ above.

1.4. Contribution. We show that whenever $(\sigma_d)_{d\in\mathbb{N}}$ and $(\mu_d)_{d\in\mathbb{N}}$ are affine functions (this includes the important case of the Black–Scholes equation in option pricing) and the initial values $(\varphi_d)_{d\in\mathbb{N}}$ can be approximated by deep neural networks without the curse of dimensionality (this can easily shown to be true for a large number of relevant options such as basket call, basket put, call on max, and call on min options), there exists a polynomial $p\colon \mathbb{R}^2 \to \mathbb{R}$ such that for every $d \in \mathbb{N}$, $\varepsilon \in (0,1)$ it holds that

$$\max\{m_{d,\varepsilon}, P(\mathbf{a}_{d,\varepsilon})\} \leq p(\varepsilon^{-1}, d);$$

see Corollary 3.5. We conclude that the aforementioned deep learning–based algorithm *does not suffer from the curse of dimensionality.*

We briefly describe our proof strategy for bounding the error between the empirical risk minimizer and the end value of the Kolmogorov equation

$$\tfrac{1}{(v-u)^d} \big\| \widehat{f}_{d,m,\mathcal{H}} - F_d(T,\cdot) \big\|^2_{\mathcal{L}^2([u,v]^d)}$$

as in (1.8). By the so-called bias-variance decomposition we can represent this error as the sum of a generalization error and an approximation error, i.e.,

$$\underbrace{\mathcal{E}_d(\widehat{f}_{d,m,\mathcal{H}}) - \min_{f\in\mathcal{H}} \mathcal{E}_d(f)}_{\text{generalization error}} + \underbrace{\min_{f\in\mathcal{H}} \tfrac{1}{(v-u)^d} \big\| f - F_d(T,\cdot) \big\|^2_{\mathcal{L}^2([u,v]^d)}}_{\text{approximation error}};$$

see Lemma 2.2. Bounds on the size of neural networks with ReLU activation function

$$\rho(x) = \mathrm{ReLU}(x) := \max\{x, 0\}$$

needed to approximate $F_d(T,\cdot)$ up to a desired error have been analyzed in [29]. In Theorem 3.3 we notably extend these approximation results by proving corresponding bounds on the parameter magnitudes. This is done by analyzing the special structure of the solution to the SDE (1.7) in the case of affine coefficients σ_d and μ_d, employing the Feynman–Kac formula (1.3), and constructing a neural network simulating Monte Carlo sampling. Together with the results of [29], we achieve that neural network hypothesis classes with ReLU activation function are capable of approximating the end values $(F_d(T,\cdot))_{d\in\mathbb{N}}$ without incurring the curse of dimensionality whenever the same is true for the initial values $(\varphi_d)_{d\in\mathbb{N}}$.

We then leverage the approximation results as well as tools from [4, 18, 19] to obtain probabilistic estimates of the generalization error; see Theorem 3.4. These tools require bounds on the covering numbers[6] $\mathrm{Cov}(\mathcal{H}, r)$ of hypothesis classes consisting of neural networks. To this end, we compute the Lipschitz constant of the operator $\mathcal{F}_\rho$ which maps neural network parametrizations to the corresponding realization functions; see Theorem 2.6. Using a standard result on the covering number of balls in a Euclidean space we obtain that[7]

$$\ln \mathrm{Cov}\big(\mathcal{N}^{u,v}_{\mathrm{ReLU},\mathbf{a},R}, r\big) \leq P(\mathbf{a}) \Big[\ln\Big(\frac{4L^2 \max\{1,|u|,|v|\}}{r} \Big) + L \ln\big(R\|\mathbf{a}\|_\infty\big) \Big];$$

[6] The covering number $\mathrm{Cov}(\mathcal{H}, r)$ is the minimal number of balls of radius r covering $\mathcal{H}$; see Setting 2.3.

[7] For $p \geq 1$, a finite index set I, and $M \in \mathbb{R}^I$ we define $\|M\|_\infty := \max_{i\in I} |M_i|$ and $\|M\|_p := \big(\sum_{i\in I} |M_i|^p\big)^{1/p}$.

see Proposition 2.8. In conjunction with Hoeffding's inequality this allows us to uniformly (over the hypothesis class of neural networks) bound the error between the risk and the empirical risk. However, this requires that the regression function f_d^* as well as all functions in $\mathcal{H}$ are uniformly bounded. To that end we assume the initial value φ_d to be bounded, which by the Feynman–Kac formula (1.3) implies that also the function $f_d^* = F_d(T, \cdot)$ is bounded. Moreover, we introduce hypothesis classes of "clipped" neural networks

$$\mathcal{N}^{u,v}_{\rho,\mathbf{a},R,D} := \left\{ \mathrm{clip}_D \circ g \colon g \in \mathcal{N}^{u,v}_{\rho,\mathbf{a},R} \right\},$$

where $\mathrm{clip}_D := \min\{|\cdot|, D\}\,\mathrm{sgn}(\cdot)$ denotes a clipping function with clipping amplitude D. This can be interpreted as incorporating the prior knowledge about the boundedness of the regression function into our hypothesis class. In Appendix A.4 we show that the clipping function can be represented as a small ReLU network so that clipped ReLU networks are in fact standard neural networks.

Note that there exist different concepts and known results in order to bound the generalization error (see, for instance, [4, 6, 8, 9, 27, 50]). The present paper intends to stress the interplay between the approximation and generalization error and gives a complete proof in order to rigorously show the absence of the curse of dimensionality for our particular problem.

We are now ready to formulate a first specific result of this paper as an appetizer. It demonstrates that deep learning–based ERM succeeds in solving the option pricing problem for European put options without incurring the curse of dimensionality.

Theorem 1.1 (pricing of options without curse of dimensionality). *Let $T, K, D \in [1, \infty)$, $u \in \mathbb{R}$, and $v \in (u, \infty)$, and let $(\Omega, \mathcal{G}, \mathbb{P}, (\mathcal{G}_t))$ be a filtered probability space. For every dimension $d \in \mathbb{N}$ let $c_d \in [-D, D]^d$, let the initial value $\varphi_d \in \mathcal{C}(\mathbb{R}^d, \mathbb{R})$ satisfy for every $x \in \mathbb{R}^d$ that*

$$\varphi_d(x) = \min\left\{ \max\left\{ D - c_d \cdot x, 0 \right\}, D \right\},$$

let the drift and diffusion coefficients $\mu_d \in \mathcal{C}(\mathbb{R}^d, \mathbb{R}^d)$, $\sigma_d \in \mathcal{C}(\mathbb{R}^d, \mathbb{R}^{d\times d})$ be affine functions satisfying for every $x \in \mathbb{R}^d$ that

$$\|\sigma_d(x)\|_2 + \|\mu_d(x)\|_2 \leq K(1 + \|x\|_2),$$

and let $F_d \in \mathcal{C}([0,T] \times \mathbb{R}^d, \mathbb{R})$ be the unique at most polynomially growing viscosity solution[8] of the corresponding d-dimensional Kolmogorov equation

$$\begin{cases} \frac{\partial F_d}{\partial t}(t,x) = \frac{1}{2}\mathrm{Trace}\big(\sigma_d(x)[\sigma_d(x)]^*(\mathrm{Hess}_x F_d)(t,x)\big) + \mu_d(x) \cdot (\nabla_x F_d)(t,x) \\ F_d(0,x) = \varphi_d(x). \end{cases}$$

For every $d \in \mathbb{N}$ let the input data $X_d \sim \mathcal{U}([u,v]^d)$ be uniformly distributed on $[u,v]^d$ and $\mathcal{G}_0$-measurable, let B^d be a d-dimensional $(\mathcal{G}_t)$-Brownian motion, let $(S_t^{X_d})_{t\in[0,T]}$ be a $(\mathcal{G}_t)$-adapted stochastic process with continuous sample paths satisfying the SDE

$$dS_t^{X_d} = \sigma_d(S_t^{X_d})dB_t^d + \mu_d(S_t^{X_d})dt \quad \text{and} \quad S_0^{X_d} = X_d$$

[8] We refer the interested reader to [32] for the definition and properties of viscosity solutions.

$\mathbb{P}$-a.s. for every $t \in [0,T]$, define the label $Y_d := \varphi_d(S_T^{X_d})$, and let $((X_d^{(i)}, Y_d^{(i)}))_{i\in\mathbb{N}}$ be i.i.d. random variables (training data) with $(X_d^{(1)}, Y_d^{(1)}) \sim (X_d, Y_d)$. Then there exists a constant $C \in (0,\infty)$ such that the following holds: For every $d, m \in \mathbb{N}$, $\varepsilon, \varrho \in (0,1)$ with

$$m \geq Cd\varepsilon^{-4}\big(1 + \ln(d\varepsilon^{-1}\varrho^{-1})\big) \quad \textit{(number of samples)}$$

there exist $\mathbf{a} = (d, a_1, a_2, 1) \in \mathbb{N}^4$ and $R \in [1,\infty)$ such that it holds that

(i) $\mathbb{P}\left[\frac{1}{(v-u)^d}\big\|\widehat{f}_{d,m,\mathcal{H}} - F_d(T,\cdot)\big\|^2_{\mathcal{L}^2([u,v]^d)} \leq \varepsilon\right] \geq 1 - \varrho$,

(ii) $P(\mathbf{a}) \leq Cd\varepsilon^{-2}$ *(number of parameters),*

(iii) $R \leq Cd^{3/2}\varepsilon^{-1}$ *(parameter bound), and*

(iv) $\max\{a_1, a_2\} \leq Cd\varepsilon^{-1}$ *(size of the architecture),*

where $\widehat{f}_{d,m,\mathcal{H}} \in \operatorname{argmin}_{f\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^m \big(f(X_d^{(i)}) - Y_d^{(i)}\big)^2$ is a measurable empirical risk minimizer in the corresponding hypothesis class of clipped ReLU networks $\mathcal{H} := \mathcal{N}^{u,v}_{\mathrm{ReLU},\mathbf{a},R,D}$.

A proof will be given in Subsection 3.3. In a more general context, Theorem 3.4 states that a result analogous to Theorem 1.1 holds true whenever the initial values $(\varphi_d)_{d\in\mathbb{N}}$ can be approximated by ReLU networks without the curse of dimensionality.

Note that our analysis does not consider the computational cost of solving the nonsmooth, nonconvex ERM problem (1.4). This is typically achieved by stochastic first order optimization methods whose theoretical analysis is still an open problem. While there are many interesting approaches to the latter question, they tend to require very strong assumptions (e.g., (almost) linearity, convexity, extreme overparametrization, or inverse stability of $\mathcal{F}_\rho$ [2, 13, 17, 20, 39, 45, 46, 48, 59]), which we want to avoid in our analysis.

1.5. Extensions. Our results in Section 3, in conjunction with the results of [29], can be applied to prove the absence of the curse of dimensionality in the pricing of (capped) basket call, basket put, call on max, and call on min options. Moreover, the results of Section 2 hold for a general statistical learning problem within Setting 2.1. That is, ReLU network approximation results for the regression function translate directly into generalization results without incurring the curse of dimensionality. If suitable learning problems can be established, this work can extend various neural network approximation results for PDEs (see, e.g., [30, 42, 56]) to also consider the generalization error and get one step closer to a full error analysis. For instance, there are stronger approximation results for more restricted option pricing problems [23], and there are very recent approximation results for semilinear heat equations [37] and Kolmogorov equations with (time-inhomogeneous) nonlinear coefficients [38, 55] where the dependence on the dimension is polynomial. Using a generalized version of Lemma 3.2, the findings of this paper can be used to prove that the corresponding ERM problem achieves, with high probability, a desired accuracy ε with the number of samples and the size of the hypothesis class scaling only polynomially in d and ε^{-1}. In particular this means that the presented methods are not restricted to the case of linear Kolmogorov equations with affine drift and diffusion coefficients. Finally, note that one obtains similar results for any continuous piecewise linear activation function with a finite number of breakpoints; see the comment after Theorem 2.6 and [62, Proposition 1].

1.6. Outline. The outline is as follows. In Section 2 we present our main result related to the generalization of neural networks in a rather general setting. Whenever the regression functions $(f_d^*)_{d\in\mathbb{N}}$ can be approximated without the curse of dimensionality by clipped ReLU networks, we show that also the number m of required training samples to achieve a desired accuracy ε with high probability does not suffer from the curse of dimensionality. This result is proven using tools from statistical learning theory and covering number estimates of neural network hypothesis classes. In Section 3 we reformulate the numerical approximation of $F_d(T,\cdot)$ on $[u,v]^d$ as a statistical learning problem and extend a result of [29] claiming that the end values $(F_d(T,\cdot))_{d\in\mathbb{N}}$ can be approximated by clipped ReLU networks without the curse of dimensionality. Therefore our results from Section 2 apply and give rise to quantitative polynomial bounds on the number of samples and the size of the network in Theorem 3.4.

2. Results in statistical learning theory.

The present section develops generalization bounds for ERM problems in the spirit of [4, 18, 19].

2.1. A generalization result based on covering numbers.

Setting 2.1 describes a standard statistical learning problem.

Setting 2.1 (statistical learning problem). Let $u\in\mathbb{R}$, $v\in(u,\infty)$, and $D\in[1,\infty)$, and let $(\Omega,\mathcal{G},\mathbb{P})$ be a probability space. For every $d\in\mathbb{N}$ let

$$X_d:\Omega\to[u,v]^d \quad (\textit{input data}) \quad\text{and}\quad Y_d:\Omega\to[-D,D] \quad (\textit{label})$$

be random variables, let $\mathbb{P}_{X_d}$ be the image measure of X_d on the hypercube $[u,v]^d$, let

$$(X_d^{(i)},Y_d^{(i)})\colon\Omega\to[u,v]^d\times[-D,D],\quad i\in\mathbb{N},\quad (\textit{training data})$$

be i.i.d. random variables with $(X_d^{(1)},Y_d^{(1)})\sim(X_d,Y_d)$, and let $f_d^*\in L^2(\mathbb{P}_{X_d})$ satisfy[9] for $\mathbb{P}_{X_d}$-a.s. $x\in[u,v]^d$ that

$$f_d^*(x)=\mathbb{E}\left[Y_d\middle|X_d=x\right] \quad (\textit{regression function}).$$

For every $d,m\in\mathbb{N}$ and Borel measurable function $f\colon[u,v]^d\to\mathbb{R}$ define the *risk* $\mathcal{E}_d(f)\in[0,\infty]$ and the *empirical risk* $\widehat{\mathcal{E}}_{d,m}(f)\colon\Omega\to[0,\infty)$ by

$$\mathcal{E}_d(f):=\mathbb{E}\Big[\big(f(X_d)-Y_d\big)^2\Big] \quad\text{and}\quad \widehat{\mathcal{E}}_{d,m}(f):=\tfrac{1}{m}\sum_{i=1}^m\big(f(X_d^{(i)})-Y_d^{(i)}\big)^2.$$

For every $d,m\in\mathbb{N}$ and every compact $\mathcal{H}\subseteq\mathcal{C}([u,v]^d,\mathbb{R})$ (*hypothesis class*) let

$$f_{d,\mathcal{H}}\in\operatorname*{argmin}_{f\in\mathcal{H}}\mathcal{E}_d(f) \quad (\textit{best approximation}), \tag{2.1}$$

and for every $d,m\in\mathbb{N}$, $\omega\in\Omega$ and every compact $\mathcal{H}\subseteq\mathcal{C}([u,v]^d,\mathbb{R})$ let

$$\widehat{f}_{d,m,\mathcal{H}}(\omega)\in\operatorname*{argmin}_{f\in\mathcal{H}}\widehat{\mathcal{E}}_{d,m}(f)(\omega) \quad (\textit{empirical regression function}) \tag{2.2}$$

[9] We define the Hilbert space $\mathcal{L}^2(\mathbb{P}_{X_d})$ as the space of all Borel measurable functions $f\colon[u,v]^d\to\mathbb{R}$ with finite norm $\|f\|_{\mathcal{L}^2(\mathbb{P}_{X_d})}=\big(\int_{[u,v]^d}f^2\,d\mathbb{P}_{X_d}\big)^{1/2}<\infty$ where functions coinciding $\mathbb{P}_{X_d}$-a.s. are identified as usual.

such that the mapping $\Omega \ni \omega \mapsto \widehat{f}_{d,m,\mathcal{H}}(\omega)$ is measurable.

We want to emphasize that the minima in (2.1) and (2.2) will be attained due to the compactness of our hypothesis class, but they need not be unique. We require the mapping $\Omega \ni \omega \mapsto \widehat{f}_{d,m,\mathcal{H}}(\omega)$ to be measurable in order to view the risk of the empirical regression function as a random variable $\Omega \ni \omega \mapsto \mathcal{E}_d\big(\widehat{f}_{d,m,\mathcal{H}}(\omega)\big)$. This ensures that the probability in the generalization error bound (Theorem 2.4) is well-defined. While this technical assumption is often not explicitly stated in the literature on statistical learning theory it is actually crucial for analyzing the generalization error. In our setting (by choosing a suitable minimizer) measurability can indeed be satisfied; see Appendix A.1.

The following lemma states that the regression function f_d^* indeed minimizes the risk and the function $f_{d,\mathcal{H}}$ is a best approximation of f_d^* in $\mathcal{H}$ with respect to the $\mathcal{L}^2(\mathbb{P}_{X_d})$-norm. Moreover, it offers a decomposition of the error between the empirical regression function $\widehat{f}_{d,m,\mathcal{H}}$ and the regression function f_d^*, often referred to as the bias-variance decomposition.

Lemma 2.2 (bias-variance decomposition). *Assume Setting 2.1. Let $d, m \in \mathbb{N}$ and let $\mathcal{H} \subseteq \mathcal{C}([u,v]^d, \mathbb{R})$ be compact. Then it holds that*

(i) $\mathcal{E}_d\big(f_d^*\big) = \min_{f \in \mathcal{L}^2(\mathbb{P}_{X_d})} \mathcal{E}_d\big(f\big)$,

(ii) $\big\|f_{d,\mathcal{H}} - f_d^*\big\|_{\mathcal{L}^2(\mathbb{P}_{X_d})} = \min_{f \in \mathcal{H}} \big\|f - f_d^*\big\|_{\mathcal{L}^2(\mathbb{P}_{X_d})}$, *and*

(iii) $$\big\|\widehat{f}_{d,m,\mathcal{H}} - f_d^*\big\|^2_{\mathcal{L}^2(\mathbb{P}_{X_d})} = \underbrace{\mathcal{E}_d\big(\widehat{f}_{d,m,\mathcal{H}}\big) - \mathcal{E}_d\big(f_{d,\mathcal{H}}\big)}_{\text{generalization error (variance)}} + \underbrace{\big\|f_{d,\mathcal{H}} - f_d^*\big\|^2_{\mathcal{L}^2(\mathbb{P}_{X_d})}}_{\text{approximation error (bias)}}.$$

For a corresponding result see also [18]. The proof in Appendix A.2 is based on the fact that we consider the square loss, and thus the risk of $f \in \mathcal{L}^2(\mathbb{P}_{X_d})$ can be represented as

$$\mathcal{E}_d(f) = \big\|f - f_d^*\big\|^2_{\mathcal{L}^2(\mathbb{P}_{X_d})} + \mathcal{E}_d(f_d^*).$$

We now introduce the concept of covering numbers in order to bound the generalization error.

Setting 2.3 (covering number). For every $r \in (0,1)$, every normed vector space $(\mathcal{Z}, \|\cdot\|)$, and every compact subset $\mathcal{H} \subseteq \mathcal{Z}$ we define the *r-covering number of $\mathcal{H}$ w.r.t. $\|\cdot\|$* by

$$\mathrm{Cov}(\mathcal{H}, \|\cdot\|, r) := \min\Big\{n \in \mathbb{N} : \text{ There exists } (f_i)_{i=1}^n \subseteq \mathcal{H} \text{ with } \mathcal{H} \subseteq \bigcup_{i=1}^n \mathit{Ball}_r(f_i)\Big\},$$

where $\mathit{Ball}_r(f) := \{g \in \mathcal{H} : \|f - g\| \leq r\}$ denotes the *ball of radius r around $f \in \mathcal{H}$*. If the norm is clear from the context, we will use the abbreviation $\mathrm{Cov}(\mathcal{H}, r) := \mathrm{Cov}(\mathcal{H}, \|\cdot\|, r)$.

Assume that the functions in our hypothesis class $\mathcal{H}$ are uniformly bounded and that balls of radius r around the functions $(f_i)_{i=1}^n$ cover $\mathcal{H}$. We can then use the (uniform) Lipschitz continuity of the (empirical) risk to bound the generalization error by

$$\begin{aligned}\mathcal{E}_d\big(\widehat{f}_{d,m,\mathcal{H}}\big) - \mathcal{E}_d\big(f_{d,\mathcal{H}}\big) &\leq \mathcal{E}_d\big(\widehat{f}_{d,m,\mathcal{H}}\big) - \widehat{\mathcal{E}}_{d,m}\big(\widehat{f}_{d,m,\mathcal{H}}\big) + \widehat{\mathcal{E}}_{d,m}\big(f_{d,\mathcal{H}}\big) - \mathcal{E}_d\big(f_{d,\mathcal{H}}\big) \\ &\leq 2r\big[\,\mathrm{Lip}(\mathcal{E}_d) + \mathrm{Lip}(\widehat{\mathcal{E}}_{d,m})\big] + 2\max_{i=1}^n \Big|\mathcal{E}_d(f_i) - \widehat{\mathcal{E}}_{d,m}(f_i)\Big|\end{aligned}$$

and employ Hoeffding's inequality and a union bound to obtain the following estimate.

Theorem 2.4 (generalization error bound). *Assume Settings 2.1 and 2.3. Let $\varepsilon \in (0,1)$, $d, m \in \mathbb{N}$ and let $\mathcal{H} \subseteq \mathcal{C}([u,v]^d, \mathbb{R})$ be compact with $\sup_{f\in\mathcal{H}} \|f\|_{\mathcal{L}^\infty} \leq D$. Then it holds that*

$$\mathbb{P}\left[\mathcal{E}_d(\widehat{f}_{d,m,\mathcal{H}}) - \mathcal{E}_d(f_{d,\mathcal{H}}) \leq \varepsilon\right] \geq 1 - 2\mathrm{Cov}\Big(\mathcal{H}, \frac{\varepsilon}{32D}\Big) \exp\left(-\frac{m\varepsilon^2}{128D^4}\right).$$

This result is adapted from [4, Theorem 17.1], [19, Theorem 3.14], and [49, Example 3.31], and for the sake of completeness we present a detailed proof in Appendix A.3.

2.2. Covering numbers of neural network hypothesis classes. As a natural next step we prove estimates on the covering numbers of neural network hypothesis classes in order to leverage the result of Theorem 2.4. Note that for different assumptions (i.e., boundedness assumptions on the activation function, different norms on the parameters, or evaluation of the neural networks on input data) similar approaches can be found in [4, 8].

The following setting describes suitable hypothesis classes based on neural networks. From now on we only consider neural networks with ReLU activation function and therefore omit writing the index $\rho = \mathrm{ReLU}$ in our notation.

Setting 2.5 (neural networks). Assume Setting 2.1. For every $k, n \in \mathbb{N}$, $W \in \mathbb{R}^{k\times n}$, $B \in \mathbb{R}^k$ let $\mathcal{A}_{W,B} \in \mathcal{C}(\mathbb{R}^n, \mathbb{R}^k)$ be the affine mapping which satisfies for every $x \in \mathbb{R}^n$ that $\mathcal{A}_{W,B}(x) := Wx + B$. For every $n \in \mathbb{N}$, $x = (x_i)_{i=1}^n \in \mathbb{R}^n$ we define

$$\mathrm{ReLU}_*(x) := \big(\max\{x_i, 0\}\big)_{i=1}^n \quad (\textit{componentwise rectified linear unit}).$$

For every $L \in \mathbb{N}$, $\mathbf{a} = (a_0, a_1, \ldots, a_L) \in \mathbb{N}^{L+1}$ (*network architecture*) we define

$$\mathcal{P}_{\mathbf{a}} := \bigtimes_{l=1}^{L} \big(\mathbb{R}^{a_l \times a_{l-1}} \times \mathbb{R}^{a_l}\big) \quad (\textit{set of neural network parametrizations}),$$

$L(\mathbf{a}) := L$ (*number of layers*), and

$$P(\mathbf{a}) := \sum_{l=1}^{L} a_l a_{l-1} + a_l \quad (\textit{number of parameters}).$$

For every $L \in \mathbb{N}$, $\mathbf{a} \in \mathbb{N}^{L+1}$, $\boldsymbol{\theta} = ((W_l, B_l))_{l=1}^L \in \mathcal{P}_{\mathbf{a}}$ we define the *neural network realization function* $\mathcal{F}(\boldsymbol{\theta}) \in \mathcal{C}(\mathbb{R}^{a_0}, \mathbb{R}^{a_L})$ by

$$\mathcal{F}(\boldsymbol{\theta}) := \mathcal{A}_{W_L,B_L} \circ \mathrm{ReLU}_* \circ \mathcal{A}_{W_{L-1},B_{L-1}} \circ \mathrm{ReLU}_* \circ \cdots \circ \mathrm{ReLU}_* \circ \mathcal{A}_{W_1,B_1}$$

and its restriction $\mathcal{F}^{u,v}(\boldsymbol{\theta}) := \mathcal{F}(\boldsymbol{\theta})|_{[u,v]^{a_0}} \in \mathcal{C}([u,v]^{a_0}, \mathbb{R}^{a_L})$ to the hypercube $[u,v]^{a_0}$. For every $d \in \mathbb{N}$ let the *admissible network architectures* be given by

$$\mathbf{A}_d := \bigcup_{L\in\mathbb{N}} \big\{(a_0, a_1, \ldots, a_L) \in \mathbb{N}^{L+1} : \ a_0 = d, \ a_L = 1\big\}.$$

For every $d \in \mathbb{N}$, $\mathbf{a} \in \mathbf{A}_d$, $R \in (0, \infty)$ (*parameter bound*) define the *set of bounded neural network parametrizations*

$$\mathcal{P}_{\mathbf{a},R} := \bigtimes_{l=1}^{L} \left([-R, R]^{a_l \times a_{l-1}} \times [-R, R]^{a_l} \right) = \{ \boldsymbol{\theta} \in \mathcal{P}_a \colon \|\boldsymbol{\theta}\|_\infty \leq R \},$$

the *hypothesis class of neural networks*

$$\mathcal{N}^{u,v}_{\mathbf{a},R} := \mathcal{F}^{u,v}(\mathcal{P}_{\mathbf{a},R}) = \left\{ \Big([u,v]^d \ni x \mapsto \mathcal{F}(\boldsymbol{\theta})(x) \Big) \colon \boldsymbol{\theta} \in \mathcal{P}_{\mathbf{a},R} \right\},$$

and the *hypothesis class of clipped neural networks*

$$\mathcal{N}^{u,v}_{\mathbf{a},R,D} := \left\{ \mathrm{clip}_D \circ g \colon g \in \mathcal{N}^{u,v}_{\mathbf{a},R} \right\},$$

where $\mathrm{clip}_D \in \mathcal{C}(\mathbb{R}, \mathbb{R})$ (*clipping function*) satisfies for every $x \in \mathbb{R}$ that

$$\mathrm{clip}_D(x) := \min\{|x|, D\} \operatorname{sgn}(x).$$

The hypothesis classes $\mathcal{N}^{u,v}_{\mathbf{a},R,D}$ are somewhat nonstandard in the sense that the clipping function clip_D is applied to the output of a neural network realization. The reason for our choice of this definition is that Theorem 2.4 requires that the set of neural networks over which the ERM problem is solved consists of uniformly bounded functions. In Appendix A.4 we show that the clipping function clip_D can be represented as a small neural network, which implies that the seemingly nonstandard classes $\mathcal{N}^{u,v}_{\mathbf{a},R,D}$ are actually conventional neural network classes that can be trained with standard methods [35, 40, 43, 51].

The next theorem quantifies the Lipschitz continuity $\mathrm{Lip}(\mathcal{F}^{u,v})$ of the operator

$$\mathcal{F}^{u,v} \colon (\mathcal{P}_{\mathbf{a},R}, \|\cdot\|_\infty) \to (\mathcal{N}^{u,v}_{\mathbf{a},R}, \|\cdot\|_{\mathcal{L}^\infty})$$

which maps bounded neural network parametrizations with fixed architecture to the corresponding realization functions (restricted to $[u,v]^d$).

Theorem 2.6 (Lipschitz continuity of $\mathcal{F}$). *Assume Setting 2.5. Let $d \in \mathbb{N}$, $\mathbf{a} \in \mathbf{A}_d$, and $R \in [1, \infty)$. Then for every $\boldsymbol{\theta}, \boldsymbol{\eta} \in \mathcal{P}_{\mathbf{a},R}$ it holds that*

$$\|\mathcal{F}^{u,v}(\boldsymbol{\theta}) - \mathcal{F}^{u,v}(\boldsymbol{\eta})\|_{\mathcal{L}^\infty} \leq 2 \max\{1, |u|, |v|\} L(\mathbf{a})^2 R^{L(\mathbf{a})-1} \|\mathbf{a}\|_\infty^{L(\mathbf{a})} \|\boldsymbol{\theta} - \boldsymbol{\eta}\|_\infty.$$

The proof is based on estimating the error amplification in each layer of the neural network and can be found in Appendix A.5. Similar results can be established for any Lipschitz continuous activation function; see also [53] for a general nonquantitative result. Note that Theorem 2.6 in particular implies that $\mathcal{N}^{u,v}_{\mathbf{a},R}$ and $\mathcal{N}^{u,v}_{\mathbf{a},R,D}$ are compact subsets of $\mathcal{C}([u,v]^d, \mathbb{R})$ and thus valid hypothesis classes, as required by Setting 2.1. Next, we recall a basic result on the covering number of a hypercube w.r.t. the maximum norm $\|\cdot\|_\infty$. Note that a similar statement holds for any ball in a finite-dimensional Banach space; see [18, Proposition 5].

Lemma 2.7 (covering numbers of balls). *Assume Setting 2.3. Let $n \in \mathbb{N}$, $R \in [1, \infty)$, and $r \in (0, 1)$, and define $Ball_R := \{\boldsymbol{\theta} \in \mathbb{R}^n : \|\boldsymbol{\theta}\|_\infty \leq R\}$. Then it holds that*

$$\ln \mathrm{Cov}(Ball_R, \|\cdot\|_\infty, r) \leq n \ln\left[\frac{R}{r}\right].$$

Proof of Lemma 2.7. The claim follows by a simple counting argument. ∎

Together with Theorem 2.6 this allows us to bound the covering number of our hypothesis class of (clipped) neural networks.

Proposition 2.8 (covering numbers of neural network hypothesis classes). *Assume Settings 2.3 and 2.5. Let $d \in \mathbb{N}$, $\mathbf{a} \in \mathbf{A}_d$, $r \in (0, 1)$, and $R \in [1, \infty)$. Then it holds that*

$$\begin{aligned}\ln \mathrm{Cov}\big(\mathcal{N}^{u,v}_{\mathbf{a},R,D}, r\big) &\leq \ln \mathrm{Cov}\big(\mathcal{N}^{u,v}_{\mathbf{a},R}, r\big) \\ &\leq P(\mathbf{a})\Big[\ln\Big(\frac{4L(\mathbf{a})^2 \max\{1, |u|, |v|\}}{r}\Big) + L(\mathbf{a}) \ln\big(R\|\mathbf{a}\|_\infty\big)\Big].\end{aligned}$$

The proof in Appendix A.6 is based on the behavior of covering numbers under the action of a Lipschitz function, i.e.,

$$\mathrm{Cov}\big(\mathcal{F}^{u,v}(\mathcal{P}_{\mathbf{a},R}), r\big) \leq \mathrm{Cov}\Big(\mathcal{P}_{\mathbf{a},R}, \frac{r}{\mathrm{Lip}(\mathcal{F}^{u,v})}\Big),$$

and uses the facts that the clipping function is nonexpansive, i.e., $\mathrm{Lip}(\mathrm{clip}_D) = 1$, and that $\mathcal{P}_{\mathbf{a},R} \simeq \{\boldsymbol{\theta} \in \mathbb{R}^{P(\mathbf{a})} : \|\boldsymbol{\theta}\|_\infty \leq R\}$.

2.3. Analysis of the generalization error. Combining Theorem 2.4 and Proposition 2.8, the following theorem describes our main result related to the generalization capabilities of hypothesis classes consisting of clipped ReLU networks.

Theorem 2.9 (neural network generalization error bound). *Assume Setting 2.5. Let $h \in \mathcal{C}((0, \infty)^5, \mathbb{R})$ satisfy for every $x = (x_i)_{i=1}^5 \in (0, \infty)^5$ that*

$$h(x) = 128 D^4 x_1^2 \Big[\ln(2) + x_2 + x_3 x_4 x_5 + x_4 \ln\big(128 D \max\{1, |u|, |v|\} x_1 x_5^2\big)\Big],$$

let $d, m \in \mathbb{N}$, $\varepsilon, \varrho \in (0, 1)$, $\mathbf{a} \in \mathbf{A}_d$, $R \in [1, \infty)$ with

$$m \geq h\left(\varepsilon^{-1}, \ln(\varrho^{-1}), \ln(R\|\mathbf{a}\|_\infty), P(\mathbf{a}), L(\mathbf{a})\right),$$

and define $\mathcal{H} := \mathcal{N}^{u,v}_{\mathbf{a},R,D}$. Then it holds that

$$\mathbb{P}\left[\mathcal{E}_d\big(\widehat{f}_{d,m,\mathcal{H}}\big) - \mathcal{E}_d\big(f_{d,\mathcal{H}}\big) \leq \varepsilon\right] \geq 1 - \varrho.$$

Proof of Theorem 2.9. Proposition 2.8 implies that

$$\begin{aligned} m &\geq h\left(\varepsilon^{-1}, \ln(\varrho^{-1}), \ln(R\|\mathbf{a}\|_\infty), P(\mathbf{a}), L(\mathbf{a})\right) \\ &= 128 D^4 \varepsilon^{-2}\Big[\ln(2\varrho^{-1}) + P(\mathbf{a})\Big(\ln\left(128 D \max\{1, |u|, |v|\} \varepsilon^{-1} L(\mathbf{a})^2\right) + L(\mathbf{a}) \ln\big(R\|\mathbf{a}\|_\infty\big)\Big)\Big] \\ &\geq 128 D^4 \varepsilon^{-2}\left[\ln(2\varrho^{-1}) + \ln \mathrm{Cov}\Big(\mathcal{H}, \frac{\varepsilon}{32 D}\Big)\right].\end{aligned}$$

Now Theorem 2.4 and a simple calculation ensures that

$$\mathbb{P}\left[\mathcal{E}_d(\widehat{f}_{d,m,\mathcal{H}}) - \mathcal{E}_d(f_{d,\mathcal{H}}) \leq \varepsilon\right] \geq 1 - 2\mathrm{Cov}\left(\mathcal{H}, \frac{\varepsilon}{32D}\right) \exp\left(-\frac{m\varepsilon^2}{128D^4}\right) \geq 1 - \varrho,$$

and this proves the theorem. ∎

Next we show how Theorem 2.9 can be used to leverage bounds on the approximation error in order to obtain quantitative bounds on the generalization error.

Corollary 2.10 (approximation implies generalization). *Assume Setting 2.5. Let $d \in \mathbb{N}$, $\varepsilon \in (0,1)$, $\mathbf{a} \in \mathbf{A}_d$, $R \in [1,\infty)$, and $g \in \mathcal{H} := \mathcal{N}^{u,v}_{\mathbf{a},R,D}$ with*

$$\left\|g - f_d^*\right\|^2_{\mathcal{L}^2(\mathbb{P}_{X_d})} \leq \varepsilon/2,$$

let $h \in \mathcal{C}((0,\infty)^5, \mathbb{R})$ be given as in Theorem 2.9, and let $m \in \mathbb{N}$, $\varrho \in (0,1)$ with

$$m \geq h\left(2\varepsilon^{-1}, \ln(\varrho^{-1}), \ln(R\|\mathbf{a}\|_\infty), P(\mathbf{a}), L(\mathbf{a})\right).$$

Then it holds that

$$\mathbb{P}\left[\left\|\widehat{f}_{d,m,\mathcal{H}} - f_d^*\right\|^2_{\mathcal{L}^2(\mathbb{P}_{X_d})} \leq \varepsilon\right] \geq 1 - \varrho.$$

Proof of Corollary 2.10. Lemma 2.2 ensures that

$$\left\|f_{d,\mathcal{H}} - f_d^*\right\|^2_{\mathcal{L}^2(\mathbb{P}_{X_d})} \leq \left\|g - f_d^*\right\|^2_{\mathcal{L}^2(\mathbb{P}_{X_d})} \leq \varepsilon/2$$

and hence $\left\|\widehat{f}_{d,m,\mathcal{H}} - f_d^*\right\|^2_{\mathcal{L}^2(\mathbb{P}_{X_d})} \leq \mathcal{E}_d(\widehat{f}_{d,m,\mathcal{H}}) - \mathcal{E}_d(f_{d,\mathcal{H}}) + \varepsilon/2$. Theorem 2.9 (with $\varepsilon \leftarrow \varepsilon/2$) now directly implies the desired claim. ∎

The previous result in particular implies that whenever the family $(f_d^*)_{d\in\mathbb{N}}$ from the statistical learning problem of Setting 2.1 can be approximated by neural networks without the curse of dimensionality, then the number m of required training samples to achieve a desired accuracy with high probability does not suffer from the curse of dimensionality either. A compact version of this statement is given in the next result.

Corollary 2.11 (approximation without curse implies generalization without curse). *Assume Setting 2.5. Assume that there exists a polynomial $q : \mathbb{R}^2 \to \mathbb{R}$ such that for every $d \in \mathbb{N}$, $\varepsilon \in (0,1)$ there exist $\mathbf{a}_{d,\varepsilon} \in \mathbf{A}_d$, $R_{d,\varepsilon} \in [1,\infty)$, and $g_{d,\varepsilon} \in \mathcal{H}_{d,\varepsilon} := \mathcal{N}^{u,v}_{\mathbf{a}_{d,\varepsilon},R_{d,\varepsilon},D}$ with*

$$\max\left\{\ln(R_{d,\varepsilon}), P(\mathbf{a}_{d,\varepsilon})\right\} \leq q(d,\varepsilon^{-1}) \quad \text{and} \quad \left\|g_{d,\varepsilon} - f_d^*\right\|^2_{\mathcal{L}^2(\mathbb{P}_{X_d})} \leq \varepsilon/2.$$

Then there exists a polynomial $s : \mathbb{R}^2 \to \mathbb{R}$ such that for every $d, m \in \mathbb{N}$, $\varepsilon, \varrho \in (0,1)$ with

$$m \geq s(d,\varepsilon^{-1})(1 + \ln(\varrho^{-1}))$$

it holds that

$$\mathbb{P}\left[\left\|\widehat{f}_{d,m,\mathcal{H}_{d,\varepsilon}} - f_d^*\right\|^2_{\mathcal{L}^2(\mathbb{P}_{X_d})} \leq \varepsilon\right] \geq 1 - \varrho.$$

Proof of Corollary 2.11. Observe that for every $d \in \mathbb{N}$, $\varepsilon \in (0,1)$ it holds that

$$\max\{\ln(\|\mathbf{a}_{d,\varepsilon}\|_\infty), L(\mathbf{a}_{d,\varepsilon})\} \leq P(\mathbf{a}_{d,\varepsilon}) \leq q(d, \varepsilon^{-1})$$

and that the function $h \in \mathcal{C}((0,\infty)^5, \mathbb{R})$ from Theorem 2.9 satisfies for every $x \in (0,\infty)^5$ that

$$h(x) \leq 128 D^4 x_1^2 \Big[1 + x_4\Big(x_1 + x_3 x_5 + 2x_5 + \ln\big(128 D \max\{1, |u|, |v|\}\big) - 3\Big)\Big](1 + x_2).$$

Thus, Corollary 2.11 is a direct consequence of Corollary 2.10. ∎

3. Applications for the numerical approximation of high-dimensional PDEs. In the present section we apply the general results of Section 2 to the numerical solution of high-dimensional Kolmogorov equations.

3.1. Kolmogorov equation as learning problem. The following setting describes suitable Kolmogorov equations and the data for the corresponding statistical learning problems.

Setting 3.1 (Kolmogorov equations). Assume Setting 2.5. Let $K \in (0,\infty)$, for every $d \in \mathbb{N}$ let $\mu_d \in \mathcal{C}(\mathbb{R}^d, \mathbb{R}^d)$ (*drift coefficient*) and $\sigma_d \in \mathcal{C}(\mathbb{R}^d, \mathbb{R}^{d\times d})$ (*diffusion coefficient*) be affine functions satisfying for every $x \in \mathbb{R}^d$ that

$$\|\sigma_d(x)\|_2 + \|\mu_d(x)\|_2 \leq K(1 + \|x\|_2),$$

and let $\varphi_d \in \mathcal{C}(\mathbb{R}^d, [-D, D])$ (*initial value*). Assume that $(\varphi_d)_{d\in\mathbb{N}}$ can be approximated by neural networks in the following sense: Let $\zeta \in [1,\infty)$ and $\beta, \gamma, \kappa, \lambda, \nu \in [0,\infty)$, and let

$$\mathbf{b}_{d,\varepsilon} \in \mathbf{A}_d, \quad \boldsymbol{\eta}_{d,\varepsilon} \in \mathcal{P}_{\mathbf{b}_{d,\varepsilon}}, \quad d \in \mathbb{N},\ \varepsilon \in (0,1), \quad (\textit{neural network approximation of } \varphi_d)$$

such that for every $d \in \mathbb{N}$, $\varepsilon \in (0,1)$, $x \in \mathbb{R}^d$ it holds that[10]

(i) $|\varphi_d(x) - \mathcal{F}(\boldsymbol{\eta}_{d,\varepsilon})(x)| \leq \varepsilon(1 + \|x\|_2^\nu)$,

(ii) $|\mathcal{F}(\boldsymbol{\eta}_{d,\varepsilon})(x)| \leq D$,

(iii) $\|\boldsymbol{\eta}_{d,\varepsilon}\|_\infty \leq \zeta d^\beta \varepsilon^{-\kappa}$, and

(iv) $P(\mathbf{b}_{d,\varepsilon}) \leq \zeta d^\gamma \varepsilon^{-\lambda}$.

Let $T \in (0,\infty)$, and for every $d \in \mathbb{N}$ let $F_d \in \mathcal{C}([0,T] \times \mathbb{R}^d, \mathbb{R})$ be the unique[11] function satisfying the following:

(i) $F_d(0,x) = \varphi(x)$ for every $x \in \mathbb{R}^d$;

(ii) F_d is at most polynomially growing, i.e., there exists $\vartheta \in (0,\infty)$ such that for every $x \in \mathbb{R}^d$ it holds that $\max_{t\in[0,T]} F_d(t,x) \leq \vartheta\,(1 + \|x\|_2^\vartheta)$; and

[10] Due to the boundedness assumption on φ_d one can obtain the desired estimates in item (i) and (ii) by adapting known neural network approximation results; see [14] and Appendix A.4.

[11] For a proof see [29, Proposition 3.4(i)].

(iii) F_d is a viscosity *solution of the d-dimensional Kolmogorov equation*

$$\tfrac{\partial F_d}{\partial t}(t,x) = \tfrac{1}{2}\mathrm{Trace}\big(\sigma_d(x)[\sigma_d(x)]^*(\mathrm{Hess}_x F_d)(t,x)\big) + \mu_d(x)\cdot(\nabla_x F_d)(t,x)$$

for every $(t,x)\in(0,T)\times\mathbb{R}^d$.

Let the probability space $(\Omega,\mathcal{G},\mathbb{P})$ be equipped with a filtration $(\mathcal{G}_t)_{t\in[0,T]}$ which fulfills the usual conditions. For every $d\in\mathbb{N}$ let $(B_t^d)_{t\in[0,T]}\colon[0,T]\times\Omega\to\mathbb{R}^d$ be a d-dimensional $(\mathcal{G}_t)$-Brownian motion, and for every $\mathcal{G}_0$-measurable random variable $\chi\colon\Omega\to\mathbb{R}^d$ denote by

$$(S_t^\chi)_{t\in[0,T]}\colon[0,T]\times\Omega\to\mathbb{R}^d \quad (\textit{SDE solution process with initial value } \chi)$$

the unique $(\mathcal{G}_t)$-adapted stochastic process[12] with continuous sample paths satisfying the SDE

$$dS_t^\chi = \sigma_d(S_t^\chi)dB_t^d + \mu_d(S_t^\chi)dt \quad \text{and} \quad S_0^\chi = \chi$$

$\mathbb{P}$-a.s. for every $t\in[0,T]$. For every $d\in\mathbb{N}$ let the *input data* $X_d\colon\Omega\to[u,v]^d$ be $\mathcal{G}_0$-measurable and uniformly distributed on $[u,v]^d$ and define the *label* by $Y_d := \varphi_d(S_T^{X_d})$.

The next result shows that computing the end value $[u,v]^d\ni x\mapsto F_d(T,x)$ of the solution to the Kolmogorov equation can be restated as a learning problem.

Lemma 3.2 (Kolmogorov equation as learning problem). *Assume Setting 3.1 and let $d\in\mathbb{N}$. Then for a.e. $x\in[u,v]^d$ it holds that*

$$F_d(T,x) = f_d^*(x).$$

The result is based on work from [10] and the following formal calculation:

$$F_d(T,x) = \mathbb{E}\big[\varphi_d\left(S_T^x\right)\big] = \mathbb{E}\big[\varphi_d(S_T^{X_d})\big|X_d=x\big] = \mathbb{E}\left[Y_d\big|X_d=x\right] = f_d^*(x)$$

for a.e. $x\in[u,v]^d$; see Appendix A.7 for a rigorous proof.

3.2. Neural network generalization results for solutions of Kolmogorov equations. We first show that the end value $[u,v]^d\ni x\mapsto F_d(T,x)$ of the solution to the Kolmogorov equation can be approximated by hypothesis classes consisting of clipped ReLU networks.

Theorem 3.3 (neural network regularity result for Kolmogorov equations). *Assume Setting 3.1. Then there exist $C,c\in(0,\infty)$ such that the following holds: For every $d\in\mathbb{N}$, $\varepsilon\in(0,1)$ there exist $\mathbf{a}\in\mathbf{A}_d$, $R\in[1,\infty)$, and $g\in\mathcal{N}_{\mathbf{a},R,D}^{u,v}$ such that it holds that*

(i) $\frac{1}{(v-u)^d}\|g-F_d(T,\cdot)\|^2_{\mathcal{L}^2([u,v]^d)}\leq\varepsilon$,

(ii) $P(\mathbf{a})\leq Cd^{\nu\lambda/2+\gamma}\varepsilon^{-\lambda/2-2}$,

(iii) $R\leq Cd^{(\nu\kappa+3)/2+\beta}\varepsilon^{-\kappa/2-1}$,

(iv) $L(\mathbf{a}) = L(\mathbf{b}_{d,cd^{-\nu/2}\varepsilon^{1/2}})$, *and*

[12] The solution process is unique up to indistinguishability; see, for instance, [5, Theorem 6.2.2].

(v) $\|\mathbf{a}\|_\infty \leq C\varepsilon^{-1}\|\mathbf{b}_{d,cd^{-\nu/2}\varepsilon^{1/2}}\|_\infty$.

Except for property (iii) a similar result was shown in [29, Corollary 3.13]. We present the proof in Appendix A.8 and briefly sketch the idea in the following. First we observe that in our case of affine coefficients σ_d and μ_d there exist random variables $\mathfrak{M}$ and $\mathfrak{N}$ such that for all $x \in \mathbb{R}^d$ it holds $\mathbb{P}$-a.s. that $S_T^x = \mathfrak{M}x + \mathfrak{N}$; see Lemma A.4. Let $((\mathfrak{M}^{(j)}, \mathfrak{N}^{(j)}))_{j\in\mathbb{N}}$ be i.i.d. samples with $(\mathfrak{M}^{(1)}, \mathfrak{N}^{(1)}) \sim (\mathfrak{M}, \mathfrak{N})$. Then for fixed $x \in \mathbb{R}^d$ the mean squared error

$$\mathbb{E}\Big[\Big(F_d(T,x) - \tfrac{1}{n}\sum_{j=1}^n \mathcal{F}(\boldsymbol{\eta}_{d,\delta})\big(\mathfrak{M}^{(j)}x + \mathfrak{N}^{(j)}\big)\Big)^2\Big]$$

can be decomposed into the sum of the squared bias and the variance, i.e.,

$$\underbrace{\mathbb{E}\Big[\varphi_d(S_T^x) - \mathcal{F}(\boldsymbol{\eta}_{d,\delta})(S_T^x)\Big]^2}_{\mathcal{O}(\delta^2)} + \underbrace{\mathbb{V}\Big[\tfrac{1}{n}\sum_{j=1}^n \mathcal{F}(\boldsymbol{\eta}_{d,\delta})\big(\mathfrak{M}^{(j)}x + \mathfrak{N}^{(j)}\big)\Big]}_{\mathcal{O}(n^{-1})},$$

where we used the Feynman–Kac formula, our assumptions, and properties of Monte Carlo sampling. With more effort one can prove analogous estimates in the $\mathcal{L}^2(\mathbb{P}_{X_d})$-norm, and our setting implies that $\mathbb{P}_{X_d} = \frac{1}{(v-u)^d}\lambda_{[u,v]^d}$, where $\lambda_{[u,v]^d}$ denotes the Lebesgue measure on $[u,v]^d$. This suggests that, given $\varepsilon \in (0,1)$, for sufficient large $n \in \mathbb{N}$ and small $\delta \in (0,1)$ there exists an outcome $\omega \in \Omega$ such that with $M^{(j)} := \mathfrak{M}^{(j)}(\omega)$ and $N^{(j)} := \mathfrak{N}^{(j)}(\omega)$ it holds that

$$\tfrac{1}{(v-u)^d}\int_{[u,v]^d}\Big(F_d(T,x) - \tfrac{1}{n}\sum_{j=1}^n \mathcal{F}(\boldsymbol{\eta}_{d,\delta})\big(M^{(j)}x + N^{(j)}\big)\Big)^2 dx \leq \varepsilon.$$

In Lemma A.5 we specify a network architecture $\mathbf{a} \in \mathbf{A}_d$ and a parametrization $\boldsymbol{\theta} \in \mathcal{P}_{\mathbf{a}}$ such that for every $x \in \mathbb{R}^d$ it holds that

$$\mathcal{F}(\boldsymbol{\theta})(x) = \tfrac{1}{n}\sum_{j=1}^n \mathcal{F}(\boldsymbol{\eta}_{d,\delta})\big(M^{(j)}x + N^{(j)}\big)$$

and we bound the parameter magnitudes of $\boldsymbol{\theta}$ with the help of Lemma A.4.

Observe that the approximation result in Theorem 3.3 does not underlie the curse of dimensionality, and by Corollary 2.10 we can establish a generalization result that is free of the curse of dimensionality.

Theorem 3.4 (neural network generalization result for Kolmogorov equations). *Assume Setting 3.1 and let $h \in \mathcal{C}((0,\infty)^5, \mathbb{R})$ be given as in Theorem 2.9. Then there exist $C, c \in (0,\infty)$ such that the following holds: For every $d, m \in \mathbb{N}$, $\varepsilon, \varrho \in (0,1)$ with*

$$m \geq h\left(2\varepsilon^{-1}, \ln(\varrho^{-1}), \ln\left(R\|\mathbf{a}\|_\infty\right), P(\mathbf{a}), L(\mathbf{a})\right)$$

there exist $\mathbf{a} \in \mathbf{A}_d$ and $R \in [1,\infty)$ such that it holds that

(i) $\mathbb{P}\left[\frac{1}{(v-u)^d}\big\|\widehat{f}_{d,m,\mathcal{H}} - F_d(T,\cdot)\big\|^2_{\mathcal{L}^2([u,v]^d)} \leq \varepsilon\right] \geq 1 - \varrho,$

(ii) $P(\mathbf{a}) \leq Cd^{\nu\lambda/2+\gamma}\varepsilon^{-\lambda/2-2}$,

(iii) $R \leq Cd^{(\nu\kappa+3)/2+\beta}\varepsilon^{-\kappa/2-1}$,

(iv) $L(\mathbf{a}) = L(\mathbf{b}_{d,cd^{-\nu/2}\varepsilon^{1/2}})$, *and*

(v) $\|\mathbf{a}\|_\infty \leq C\varepsilon^{-1}\|\mathbf{b}_{d,cd^{-\nu/2}\varepsilon^{1/2}}\|_\infty$,

where $\mathcal{H} = \mathcal{N}^{u,v}_{\mathbf{a},R,D}$.

Proof of Theorem 3.4. This is a direct consequence of Theorem 3.3 (with $\varepsilon \leftarrow \varepsilon/2$) and Corollary 2.10. ∎

We can also reformulate this in a more compact form.

Corollary 3.5 (ERM for Kolmogorov equations without curse). *Assume Setting 3.1. Then there exists a polynomial* $p\colon \mathbb{R}^2 \to \mathbb{R}$ *such that the following holds: For every* $d, m \in \mathbb{N}$, $\varepsilon, \varrho \in (0,1)$ *with*

$$m \geq p(d, \varepsilon^{-1})(1 + \ln(\varrho^{-1}))$$

there exist $\mathbf{a} \in \mathbf{A}_d$ *and* $R \in [1, \infty)$ *such that it holds that*

(i) $\mathbb{P}\left[\frac{1}{(v-u)^d}\|\widehat{f}_{d,m,\mathcal{H}} - F_d(T,\cdot)\|^2_{\mathcal{L}^2([u,v]^d)} \leq \varepsilon\right] \geq 1 - \varrho$ *and*

(ii) $\max\{R, P(\mathbf{a})\} \leq p(d, \varepsilon^{-1})$,

where $\mathcal{H} = \mathcal{N}^{u,v}_{\mathbf{a},R,D}$.

Proof of Corollary 3.5. This follows directly from Theorem 3.3 and Corollary 2.11. ∎

3.3. Pricing of high-dimensional options. The proof of Theorem 1.1 from the introductory section dealing with the pricing of high-dimensional European put options is now an easy consequence of the above theory.

Proof of Theorem 1.1. We first show that the approximation of $(\varphi_d)_{d\in\mathbb{N}}$ by clipped neural networks according to Setting 3.1 is possible. Note that for every $z \in \mathbb{R}$ it holds that $\min\{z, D\} = D - \mathrm{ReLU}_*(D - z)$. This implies that for every $d \in \mathbb{N}$, $x \in \mathbb{R}^d$ it holds that

$$\varphi_d(x) = \min\left\{\max\left\{D - c_d \cdot x, 0\right\}, D\right\} = \mathcal{F}(\boldsymbol{\eta}_d)(x),$$

where

$$\boldsymbol{\eta}_d := \left(\left(-[c_d]^*, D\right), (-1, D), (-1, D)\right) \in \mathcal{P}_{(d,1,1,1),D}.$$

Accordingly, Setting 3.1 is satisfied with

$$\zeta = \max\{D, 6\}, \quad \gamma = 1, \quad \beta = \kappa = \lambda = \nu = 0, \quad \mathbf{b}_{d,\varepsilon} = (d, 1, 1, 1), \quad \boldsymbol{\eta}_{d,\varepsilon} = \boldsymbol{\eta}_d.$$

Now Theorem 3.4 and a straightforward calculation prove the claim. ∎

Appendix A. Proofs. This appendix contains various proofs and additional material omitted from the main text.

A.1. Measurability of the empirical target function. The following lemma shows that the empirical regression function can be chosen measurable as required in Setting 2.1. This implies that the risk of the empirical regression function $\Omega \ni \omega \mapsto \mathcal{E}_d\big(\widehat{f}_{d,m,\mathcal{H}}(\omega)\big)$ is measurable which is necessary for bounding the generalization error in Theorem 2.4.

Lemma A.1 (measurability of the empirical regression function). *Let $u \in \mathbb{R}$, $v \in (u, \infty)$, and $d, m \in \mathbb{N}$, let $(\Omega, \mathcal{G}, \mathbb{P})$ be a probability space, let*

$$(X_d^{(i)}, Y_d^{(i)})\colon \Omega \to [u,v]^d \times \mathbb{R}, \quad i \in \mathbb{N},$$

be random variables, and let $\mathcal{H} \subseteq \mathcal{C}([u,b]^d, \mathbb{R})$ be compact. For every $\omega \in \Omega$ one can choose

$$\widehat{f}_{d,m,\mathcal{H}}(\omega) \in \operatorname*{argmin}_{f \in \mathcal{H}} \tfrac{1}{m} \sum_{i=1}^{m} \big(f(X_d^{(i)}(\omega)) - Y_d^{(i)}(\omega)\big)^2$$

in a way, such that it holds that[13]

(i) $\Omega \ni \omega \mapsto \widehat{f}_{d,m,\mathcal{H}}(\omega)$ is $\mathcal{G}/\mathcal{B}(\mathcal{H})$-measurable and

(ii) $\Omega \ni \omega \mapsto \mathcal{E}_d\big(\widehat{f}_{d,m,\mathcal{H}}(\omega)\big)$ is $\mathcal{G}/\mathcal{B}(\mathbb{R})$-measurable.

Proof of Lemma A.1. First observe that $\mathcal{H}$ is a separable metric space induced by the uniform norm $\|\cdot\|_{\mathcal{L}^\infty}$ and that for every $f \in \mathcal{H}$ the mapping

$$\Omega \ni \omega \mapsto \widehat{\mathcal{E}}_{d,m}(f)(\omega)$$

is $\mathcal{G}/\mathcal{B}(\mathbb{R})$-measurable. By the reverse triangle inequality we obtain for every $f, g \in \mathcal{H}$ that

$$\big|\widehat{\mathcal{E}}_{d,m}(f)^{1/2} - \widehat{\mathcal{E}}_{d,m}(g)^{1/2}\big| \leq \tfrac{1}{\sqrt{m}} \big\|(f(X_d^{(i)}) - g(X_d^{(i)}))_{i=1}^m\big\|_2 \leq \|f - g\|_{\mathcal{L}^\infty}.$$

This shows that for every $\omega \in \Omega$ the function $\mathcal{H} \ni f \mapsto \widehat{\mathcal{E}}_{d,m}(f)(\omega)$ is continuous and the Measurable Maximum Theorem in [1, Theorem 18.19] ensures that the set-valued function of minimizers of

$$\min_{f \in \mathcal{H}} \widehat{\mathcal{E}}_{d,m}(f)$$

admits a measurable selector. That is to say, there exists a $\mathcal{G}/\mathcal{B}(\mathcal{H})$-measurable mapping $\widehat{f}_{d,m,\mathcal{H}}\colon \Omega \to \mathcal{H}$ such that for every $\omega \in \Omega$ it holds that

$$\widehat{f}_{d,m,\mathcal{H}}(\omega) \in \operatorname*{argmin}_{f \in \mathcal{H}} \widehat{\mathcal{E}}_{d,m}(f)(\omega).$$

[13] We denote by $\mathcal{B}(\mathcal{Z})$ the Borel σ-algebra of a topological space $\mathcal{Z}$.

This establishes item (i). For the proof of the second item observe that the risk $\mathcal{E}_d\colon \mathcal{H} \to \mathbb{R}$ is continuous and thus $\mathcal{B}(\mathcal{H})/\mathcal{B}(\mathbb{R})$-measurable. Indeed, an analogous computation as for the empirical risk above shows that for $f, g \in \mathcal{H}$ it holds that

$$\big|\mathcal{E}_d(f)^{1/2} - \mathcal{E}_d(g)^{1/2}\big| \leq \|f(X_d) - g(X_d)\|_{\mathcal{L}^2(\mathbb{P})} \leq \|f - g\|_{\mathcal{L}^\infty}.$$

This yields the claim as compositions of measurable functions are again measurable. ∎

A.2. Bias-variance decomposition.

Proof of Lemma 2.2. For every $f \in \mathcal{L}^2(\mathbb{P}_{X_d})$ it holds that

$$\begin{aligned}\mathcal{E}_d(f) &= \mathbb{E}\Big[\big(f(X_d) - f_d^*(X_d) + f_d^*(X_d) - Y_d\big)^2\Big] \\ &= \mathbb{E}\Big[\big(f(X_d) - f_d^*(X_d)\big)^2\Big] + \mathbb{E}\Big[\big(f_d^*(X_d) - Y_d\big)^2\Big] \\ &\quad + 2\mathbb{E}\big[\big(f(X_d) - f_d^*(X_d)\big)\big(f_d^*(X_d) - Y_d\big)\big]\end{aligned} \tag{A.1}$$

Observe that, due to the fact that it holds $\mathbb{P}$-a.s. that $f_d^*(X_d) = \mathbb{E}\big[Y_d\big|X_d\big]$, the tower property of the conditional expectation establishes for every $f \in \mathcal{L}^2(\mathbb{P}_{X_d})$ that

$$\begin{aligned}\mathbb{E}\big[\big(f(X_d) - f_d^*(X_d)\big)\big(f_d^*(X_d) - Y_d\big)\big] &= \mathbb{E}\Big[\mathbb{E}\Big[\big(f(X_d) - f_d^*(X_d)\big)\big)\big(f_d^*(X_d) - Y_d\big)\Big|X_d\Big]\Big] \\ &= \mathbb{E}\big[\big(f(X_d) - f_d^*(X_d)\big)\big(f_d^*(X_d) - \mathbb{E}\big[Y_d\big|X_d\big]\,\big)\big] = 0\end{aligned}$$

which, together with (A.1), implies that

$$\big\|f - f_d^*\big\|^2_{\mathcal{L}^2(\mathbb{P}_{X_d})} = \mathbb{E}\Big[\big(f(X_d) - f_d^*(X_d)\big)^2\Big] = \mathcal{E}_d(f) - \mathcal{E}_d(f_d^*). \tag{A.2}$$

This proves items (i) and (ii) and shows that it holds that

$$\big\|\widehat{f}_{d,m,\mathcal{H}} - f_d^*\big\|^2_{\mathcal{L}^2(\mathbb{P}_{X_d})} = \mathcal{E}_d(\widehat{f}_{d,m,\mathcal{H}}) - \mathcal{E}_d(f_{d,\mathcal{H}}) + \mathcal{E}_d(f_{d,\mathcal{H}}) - \mathcal{E}_d(f_d^*).$$

Finally, applying (A.2) (with $f \leftarrow f_{d,\mathcal{H}}$) proves the lemma. ∎

A.3. Bound on the generalization error.

Proof of Theorem 2.4. First note that by assumption for every $f \in \mathcal{H}$ it holds that

$$|f(X_d) - Y_d| \leq \|f\|_{\mathcal{L}^\infty} + |Y_d| \leq 2D$$

and analogously for the samples $((X_d^{(i)}, Y_d^{(i)}))_{i=1}^m$. The elementary identity

$$(y_1 - z)^2 - (y_2 - z)^2 = (y_1 - y_2)(y_1 + y_2 - 2z)$$

for real numbers $y_1, y_2, z \in \mathbb{R}$ and Jensen's inequality imply for every $f, g \in \mathcal{H}$ that

$$\big|\mathcal{E}_d(f) - \mathcal{E}_d(g)\big| \leq \mathbb{E}\big[\big|\big(f(X_d) - g(X_d)\big)\big(f(X_d) + g(X_d) - 2Y_d\big)\big|\big] \leq 4D\|f - g\|_{\mathcal{L}^\infty}$$

and

$$\begin{aligned}\left|\widehat{\mathcal{E}}_{d,m}(f)-\widehat{\mathcal{E}}_{d,m}(g)\right| &\leq \tfrac{1}{m}\sum_{i=1}^{m}\left|\left(f(X_d^{(i)})-g(X_d^{(i)})\right)\left(f(X_d^{(i)})+g(X_d^{(i)})-2Y_d^{(i)}\right)\right| \\ &\leq 4D\|f-g\|_{\mathcal{L}^\infty}.\end{aligned}$$

Now define $N := \mathrm{Cov}\left(\mathcal{H}, \|\cdot\|_{\mathcal{L}^\infty}, \frac{\varepsilon}{32D}\right)$ and choose $f_1, f_2, \ldots, f_N \in \mathcal{H}$ such that the balls

$$Ball_i := \left\{f\in\mathcal{H}\colon \|f-f_i\|_{\mathcal{L}^\infty}\leq\frac{\varepsilon}{32D}\right\}, \quad i\in\{1,2,\ldots,N\},$$

cover $\mathcal{H}$. This establishes that for every $i \in \{1,2,\ldots,N\}$, $f \in Ball_i$ it holds that

$$\begin{aligned}\left|\mathcal{E}_d(f)-\widehat{\mathcal{E}}_{d,m}(f)\right| &\leq \left|\mathcal{E}_d(f)-\mathcal{E}_d(f_i)\right|+\left|\mathcal{E}_d(f_i)-\widehat{\mathcal{E}}_{d,m}(f_i)\right|+\left|\widehat{\mathcal{E}}_{d,m}(f_i)-\widehat{\mathcal{E}}_{d,m}(f)\right| \\ &\leq 8D\|f-f_i\|_{\mathcal{L}^\infty}+\left|\mathcal{E}_d(f_i)-\widehat{\mathcal{E}}_{d,m}(f_i)\right| \\ &\leq \varepsilon/4+\left|\mathcal{E}_d(f_i)-\widehat{\mathcal{E}}_{d,m}(f_i)\right|.\end{aligned} \tag{A.3}$$

Our assumptions yield that for every $\omega \in \Omega$ it holds that

$$\begin{aligned}&\mathcal{E}_d\big(\widehat{f}_{d,m,\mathcal{H}}(\omega)\big)-\mathcal{E}_d\big(f_{d,\mathcal{H}}\big) \\ &\leq \mathcal{E}_d\big(\widehat{f}_{d,m,\mathcal{H}}(\omega)\big)-\widehat{\mathcal{E}}_{d,m}\big(\widehat{f}_{d,m,\mathcal{H}}(\omega)\big)(\omega)+\widehat{\mathcal{E}}_{d,m}\big(f_{d,\mathcal{H}}\big)(\omega)-\mathcal{E}_d\big(f_{d,\mathcal{H}}\big) \\ &\leq 2\sup_{f\in\mathcal{H}}\left|\mathcal{E}_d(f)-\widehat{\mathcal{E}}_{d,m}(f)(\omega)\right|.\end{aligned} \tag{A.4}$$

In summary (A.3) and (A.4) imply that

$$\begin{aligned}&\left\{\omega\in\Omega\colon \mathcal{E}_d\big(\widehat{f}_{d,m,\mathcal{H}}(\omega)\big)-\mathcal{E}_d\big(f_{d,\mathcal{H}}\big)\geq\varepsilon\right\} \\ &\subseteq\bigcup_{i=1}^{N}\left\{\omega\in\Omega\colon \sup_{f\in Ball_i}\left|\mathcal{E}_d(f)-\widehat{\mathcal{E}}_{d,m}(f)(\omega)\right|\geq\varepsilon/2\right\} \\ &\subseteq\bigcup_{i=1}^{N}\left\{\omega\in\Omega\colon \left|\mathcal{E}_d(f_i)-\widehat{\mathcal{E}}_{d,m}(f_i)(\omega)\right|\geq\varepsilon/4\right\}.\end{aligned} \tag{A.5}$$

Observe that for fixed $f \in \mathcal{H}$ it holds that the random variables $E_i := \big(f(X_d^{(i)})-Y_d^{(i)}\big)^2$, $i \in \{1,2,\ldots,m\}$, are independent and satisfy

$$\mathbb{E}\left[E_i\right]=\mathcal{E}_d(f), \quad \tfrac{1}{m}\sum_{i=1}^{m}E_i=\widehat{\mathcal{E}}_{d,m}(f), \quad \text{and} \quad 0\leq E_i\leq 4D^2$$

which by Hoeffding's inequality (see [36, Theorem 2]) ensures that

$$\mathbb{P}\left[\left|\mathcal{E}_d(f)-\widehat{\mathcal{E}}_{d,m}(f)\right|\geq\varepsilon/4\right]\leq 2\exp\left(-\frac{m\varepsilon^2}{128D^4}\right).$$

Together with (A.5), the monotonicity and subadditivity of the probability measure, and the measurability assumptions according to Lemma A.1 this implies that

$$\mathbb{P}\left[\mathcal{E}_d(\widehat{f}_{d,m,\mathcal{H}}) - \mathcal{E}_d(f_{d,\mathcal{H}}) \geq \varepsilon\right] \leq \sum_{i=1}^{N} \mathbb{P}\left[\left|\mathcal{E}_d(f_i) - \widehat{\mathcal{E}}_{d,m}(f_i)\right| \geq \varepsilon/4\right] \leq 2N \exp\left(-\frac{m\varepsilon^2}{128 D^4}\right).$$

Using the complement rule and plugging in the definition of N proves the theorem. ∎

A.4. Clipped neural networks are standard neural networks. We show that "clipped" neural network hypothesis classes $\mathcal{N}^{u,v}_{\mathbf{a},R,D}$ are in fact subsets of "non-clipped" ones.

Lemma A.2 (clipping function as neural network). *Assume Setting 2.5 and let*

$$\boldsymbol{\theta} := \left(\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right), \left(\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} D \\ D \end{bmatrix}\right), \left(\begin{bmatrix} -1 & 1 \end{bmatrix}, 0\right)\right) \in \mathcal{P}_{(1,2,2,1),D}.$$

Then it holds that

$$\mathrm{clip}_D = \mathcal{F}(\boldsymbol{\theta}).$$

Proof of Lemma A.2. A case distinction establishes that for every $x \in \mathbb{R}$ it holds that

$$\mathcal{F}(\boldsymbol{\theta})(x) = -\mathrm{ReLU}_*\left(D - \mathrm{ReLU}_*(x)\right) + \mathrm{ReLU}_*\left(-\mathrm{ReLU}_*(-x) + D\right) = \mathrm{clip}_D(x),$$

which proves the claim. ∎

Corollary A.3 (clipped neural networks are standard neural networks). *Assume Setting 2.5. Let $d \in \mathbb{N}$, $R \in [D, \infty)$, and $\mathbf{a} = (a_0, a_1, \ldots, a_L) \in \mathbf{A}_d$, and define*

$$\mathbf{b} := (a_0, a_1, \ldots, a_L, 2, 2, 1) \in \mathbf{A}_d.$$

Then it holds that

$$\mathcal{N}^{u,v}_{\mathbf{a},R,D} \subseteq \mathcal{N}^{u,v}_{\mathbf{b},R}.$$

Proof of Corollary A.3. The proof follows by the representation of the clipping function in Lemma A.2 and the fact that composition with a neural network does not change the magnitude of its parameters;[14] see also [54, Definition 2.2] for a formal definition. ∎

A.5. Lipschitz continuity of the realization map.

Proof of Theorem 2.6. Define $L := L(\mathbf{a})$ and $\mathfrak{m} := \max\{1, |u|, |v|\}$. We will show the following stronger statement. For every $\boldsymbol{\theta}, \boldsymbol{\eta} \in \mathcal{P}_{\mathbf{a},R}$ it holds that

$$\left\|\mathcal{F}^{u,v}(\boldsymbol{\theta}) - \mathcal{F}^{u,v}(\boldsymbol{\eta})\right\|_{\mathcal{L}^\infty} \leq \left[\mathfrak{m} L R^{L-1} \|\mathbf{a}\|_\infty^L + \sum_{l=1}^{L} l (R\|\mathbf{a}\|_\infty)^{l-1}\right] \|\boldsymbol{\theta} - \boldsymbol{\eta}\|_\infty. \tag{A.6}$$

This directly implies the statement of Theorem 2.6, as it holds that

$$\sum_{l=1}^{L} l (R\|\mathbf{a}\|_\infty)^{l-1} \leq L^2 (R\|\mathbf{a}\|_\infty)^{L-1} \leq \mathfrak{m} L^2 R^{L-1} \|\mathbf{a}\|_\infty^L.$$

[14] Because of that we did not choose the easier representation $\mathrm{clip}_D(x) = -D + \mathrm{ReLU}_*\left(2D - \mathrm{ReLU}_*(D - x)\right)$.

For the proof of (A.6) let us fix $\boldsymbol{\theta}, \boldsymbol{\eta} \in \mathcal{P}_{\mathbf{a},R}$ given by

$$\boldsymbol{\theta} = ((W_l, B_l))_{l=1}^{L} \quad \text{and} \quad \boldsymbol{\eta} = ((V_l, A_l))_{l=1}^{L}.$$

Let $r := \|\boldsymbol{\theta} - \boldsymbol{\eta}\|_\infty$, and for every $s \in \{1, \dots, L\}$ define the partial parametrizations

$$\boldsymbol{\theta}(s) = ((W_l, B_l))_{l=1}^{s}, \quad \text{and} \quad \boldsymbol{\eta}(s) = ((V_l, A_l))_{l=1}^{s},$$

the partial realization functions $f_s := \mathcal{F}^{u,v}(\boldsymbol{\theta}(s))$ and $g_s := \mathcal{F}^{u,v}(\boldsymbol{\eta}(s))$, the partial errors[15]

$$\mathfrak{e}_0 := 0 \quad \text{and} \quad \mathfrak{e}_s := \left\| f_s - g_s \right\|_{\mathcal{L}^\infty},$$

and the partial maxima

$$\mathfrak{m}_0 := \mathfrak{m} = \max\left\{1, |u|, |v|\right\} \quad \text{and} \quad \mathfrak{m}_s := \max\left\{1, \left\| f_s \right\|_{\mathcal{L}^\infty}, \left\| g_s \right\|_{\mathcal{L}^\infty}\right\}.$$

We are interested in estimating the error $\mathfrak{e}_L$ and try to bound $\mathfrak{m}_s$ relative to $\mathfrak{m}_{s-1}$ and $\mathfrak{e}_s$ relative to $\mathfrak{e}_{s-1}$. Note that for every $s \in \{2, \dots, L\}$ it holds that

$$\left\| f_s \right\|_{\mathcal{L}^\infty} = \left\| W_s \, \mathrm{ReLU}_* \left(f_{s-1} \right) + B_s \right\|_{\mathcal{L}^\infty} \leq R \|\mathbf{a}\|_\infty \mathfrak{m}_{s-1} + R.$$

Analogous computations for the case $s = 1$ and the function g_s establish that for every $s \in \{1, \dots, L\}$ it holds that $\mathfrak{m}_s \leq R\|\mathbf{a}\|_\infty \mathfrak{m}_{s-1} + R$. By induction this implies that

$$\mathfrak{m}_s \leq \mathfrak{m}(R\|\mathbf{a}\|_\infty)^s + R \sum_{l=0}^{s-1} (R\|\mathbf{a}\|_\infty)^l \tag{A.7}$$

for every $s \in \{1, \dots, L\}$. Moreover, note that for every $s \in \{2, \dots, L\}$ it holds that

$$\begin{aligned}
\mathfrak{e}_s &= \left\| \left[W_s \, \mathrm{ReLU}_* \left(f_{s-1} \right) + B_s \right] - \left[V_s \, \mathrm{ReLU}_* \left(g_{s-1} \right) + A_s \right] \right\|_{\mathcal{L}^\infty} \\
&\leq \left\| \left[W_s - V_s \right] \mathrm{ReLU}_* \left(f_{s-1} \right) \right\|_{\mathcal{L}^\infty} + \left\| V_s \left[\mathrm{ReLU}_* \left(f_{s-1} \right) - \mathrm{ReLU}_* \left(g_{s-1} \right) \right] \right\|_{\mathcal{L}^\infty} + r \\
&\leq \|\mathbf{a}\|_\infty \left(r \mathfrak{m}_{s-1} + R \mathfrak{e}_{s-1} \right) + r.
\end{aligned}$$

Together with (A.7), one proves by induction that for every $s \in \{1, 2, \dots, L\}$ it holds that

$$\mathfrak{e}_s \leq \left[\mathfrak{m} s R^{s-1} \|\mathbf{a}\|_\infty^s + \sum_{l=1}^{s} l (R\|\mathbf{a}\|_\infty)^{l-1} \right] r.$$

Setting $s = L$ proves the claim in (A.6). ∎

[15] For vector-valued functions $f \in \mathcal{C}([u,v]^d, \mathbb{R}^n)$ we define the uniform norm on $[u,v]^d$ by $\|f\|_{\mathcal{L}^\infty} = \|f\|_{\mathcal{L}^\infty([u,v]^d)} := \max_{x \in [u,v]^d} \|f(x)\|_\infty$.

A.6. Covering numbers of neural network hypothesis classes.

Proof of Proposition 2.8. To simplify the notation we define $\mathfrak{m} := \max\{1, |u|, |v|\}$,

$$\Delta := \frac{r}{2\mathfrak{m}L(\mathbf{a})^2 R^{L(\mathbf{a})-1}\|\mathbf{a}\|_\infty^{L(\mathbf{a})}}, \quad \text{and} \quad N := \mathrm{Cov}\big(\mathcal{P}_{\mathbf{a},R}, \|\cdot\|_\infty, \Delta\big).$$

Choose $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N \in \mathcal{P}_{\mathbf{a},R}$ such that for every $\boldsymbol{\theta} \in \mathcal{P}_{\mathbf{a},R}$ there exists $i \in \{1, 2, \ldots, N\}$ with $\|\boldsymbol{\theta} - \boldsymbol{\theta}_i\|_\infty \leq \Delta$, which by Theorem 2.6 and the fact that clip_D is nonexpansive implies that

$$\begin{aligned}\big\|\mathrm{clip}_D \circ \mathcal{F}^{u,v}(\boldsymbol{\theta}) - \mathrm{clip}_D \circ \mathcal{F}^{u,v}(\boldsymbol{\theta}_i)\big\|_{\mathcal{L}^\infty} &\leq \big\|\mathcal{F}^{u,v}(\boldsymbol{\theta}) - \mathcal{F}^{u,v}(\boldsymbol{\theta}_i)\big\|_{\mathcal{L}^\infty} \\ &\leq 2\mathfrak{m}L(\mathbf{a})^2 R^{L(\mathbf{a})-1}\|\mathbf{a}\|_\infty^{L(\mathbf{a})}\|\boldsymbol{\theta} - \boldsymbol{\theta}_i\|_\infty \leq r.\end{aligned}$$

Lemma 2.7 and identifying $\mathcal{P}_{\mathbf{a},R} \simeq \big\{\boldsymbol{\theta} \in \mathbb{R}^{P(\mathbf{a})} : \|\boldsymbol{\theta}\|_\infty \leq R\big\}$ hence show that

$$\begin{aligned}\ln \mathrm{Cov}\big(\mathcal{N}^{u,v}_{\mathbf{a},R,D}, \|\cdot\|_{\mathcal{L}^\infty}, r\big) \leq \ln \mathrm{Cov}\big(\mathcal{N}^{u,v}_{\mathbf{a},R}, \|\cdot\|_{\mathcal{L}^\infty}, r\big) &\leq \ln \mathrm{Cov}\big(\mathcal{P}_{\mathbf{a},R}, \|\cdot\|_\infty, \Delta\big) \\ &\leq P(\mathbf{a}) \ln\Big(\Big\lceil \frac{R}{\Delta} \Big\rceil\Big) \leq P(\mathbf{a}) \ln\Big(\frac{2R}{\Delta}\Big),\end{aligned}$$

and this proves the proposition. ∎

A.7. Kolmogorov equation as learning problem.

Proof of Lemma 3.2. The proof is based on the Feynman–Kac formula for viscosity solutions of Kolmogorov equations, which states that for every $x \in \mathbb{R}^d$ it holds that

$$F_d(T, x) = \mathbb{E}\big[\varphi_d\left(S_T^x\right)\big]; \tag{A.8}$$

see [29, Corollary 2.23(ii)]. We claim that for every $A \in \mathcal{B}([u, v]^d)$ it holds that

$$\mathbb{E}\big[\mathbb{1}_A(X_d)\varphi_d(S_T^{X_d})\big] = \int_A \mathbb{E}\left[\varphi_d(S_T^x)\right] d\mathbb{P}_{X_d}(x).$$

This would prove the lemma as it implies that for $\mathbb{P}_{X_d}$-a.s. $x \in [u, v]^d$ it holds that

$$\mathbb{E}\big[\varphi_d(S_T^{X_d})\big|X_d = x\big] = \mathbb{E}\big[\varphi_d\left(S_T^x\right)\big],$$

which by (A.8) and Setting 3.1 ensures that for a.e. $x \in [u, v]^d$ it holds that

$$f_d^*(x) = \mathbb{E}\left[Y_d\middle|X_d = x\right] = \mathbb{E}\big[\varphi_d(S_T^{X_d})\big|X_d = x\big] = \mathbb{E}\big[\varphi_d\left(S_T^x\right)\big] = F_d(T, x).$$

For the proof of the claim let us fix $A \in \mathcal{B}([u, v]^d)$ and let $g_\varepsilon \in \mathcal{C}^\infty(\mathbb{R}^d, \mathbb{R})$, $\varepsilon \in (0, 1)$, be a family of mollifiers. For every $\varepsilon \in (0, 1)$ we define the convolution with the indicator function $\mathbb{1}_{A,\varepsilon} := \mathbb{1}_A * g_\varepsilon \in \mathcal{C}^\infty(\mathbb{R}^d, \mathbb{R})$ and the continuous and bounded mapping

$$\Phi_\varepsilon : \begin{cases} \mathcal{C}([0, T], \mathbb{R}^d) & \to \quad \mathbb{R} \\ f & \mapsto \quad \mathbb{1}_{A,\varepsilon}(f(0))\varphi_d(f(T)). \end{cases}$$

In [10, Lemma 2.6(v)] it is shown that for every $\varepsilon \in (0,1)$ it holds that

$$\mathbb{E}\big[\mathbb{1}_{A,\varepsilon}(X_d)\varphi_d(S_T^{X_d})\big] = \tfrac{1}{(v-u)^d}\int_{[u,v]^d} \mathbb{1}_{A,\varepsilon}(x)\mathbb{E}\big[\varphi_d\left(S_T^x\right)\big]\,dx.$$

The fact that $\lim_{\varepsilon\to 0}\mathbb{1}_{A,\varepsilon}(x) = \mathbb{1}_A(x)$ for a.e. $x \in \mathbb{R}^d$ (see, for instance, [24, Appendix C.5]) and the dominated convergence theorem prove the claim when letting ε tend to zero. ∎

A.8. Neural network approximation result for solutions of Kolmogorov equations. The proof of Theorem 3.3 is given after the following two auxiliary lemmas. First, we show that given an SDE with affine coefficients σ_d and μ_d, its solution S_T^x also admits a (random) affine representation.

Lemma A.4 (representation of SDE solutions). *Assume Setting 3.1. Let $d \in \mathbb{N}$, let $e_i \in \mathbb{R}^d$, $i \in \{1,\dots,d\}$, be the standard basis in $\mathbb{R}^d$, for every $p, z \in [0,\infty)$ let*

$$\mathfrak{c}_p(z) := 2^{p/2}\big(z + KT + \max\{2,p\}K\sqrt{T}\big)^p \exp\big(pK^2T\big[\sqrt{T} + \max\{2,p\}\big]^2\big),$$

and define the random variables $\mathfrak{M}: \Omega \to \mathbb{R}^{d\times d}$ and $\mathfrak{N}: \Omega \to \mathbb{R}^d$ by

$$\mathfrak{M} := \big[S_T^{e_1} - S_T^0 \quad S_T^{e_2} - S_T^0 \quad \dots \quad S_T^{e_d} - S_T^0\big] \quad \text{and} \quad \mathfrak{N} := S_T^0.$$

Then for every $x \in \mathbb{R}^d$ it holds $\mathbb{P}$-a.s. that $S_T^x = \mathfrak{M}x + \mathfrak{N} = \mathcal{A}_{\mathfrak{M},\mathfrak{N}}(x)$ and it holds that[16]

(i) $\mathbb{E}\big[\,\|\mathfrak{M}\|_2 + \|\mathfrak{N}\|_2\,\big] \le 3\mathfrak{c}_1(1)d$ *and*

(ii) $\big\|\mathbb{E}\big[\,\|\mathcal{A}_{\mathfrak{M},\mathfrak{N}}\|_2^\nu\,\big]\big\|_{\mathcal{L}^2(\mathbb{P}_{X_d})} \le \mathfrak{c}_\nu(\max\{1,|u|,|v|\})d^{\nu/2}$.

Proof of Lemma A.4. A proof of the first claim can be found in [29, Lemmas 2.7 and 2.15]. For the proof of items (i) and (ii) note that for every $p \in [0,\infty)$, $x \in \mathbb{R}^d$ it holds that

$$\mathbb{E}\big[\|\mathcal{A}_{\mathfrak{M},\mathfrak{N}}(x)\|_2^p\big] = \mathbb{E}\big[\|S_T^x\|_2^p\big] \le \Big(\mathbb{E}\Big[\|S_T^x\|_2^{\max\{2,p\}}\Big]\Big)^{p/\max\{2,p\}} \le \mathfrak{c}_p(\|x\|_2);$$

see [29, Proposition 2.14]. Together with the facts that it holds that

$$\mathbb{E}\big[\,\|\mathfrak{M}\|_2 + \|\mathfrak{N}\|_2\,\big] \le \mathbb{E}\Big[\,\big\|S_T^0\big\|_2 + \sum_{i=1}^d \big\|S_T^{e_i} - S_T^0\big\|_2\Big] \le (d+1)\mathbb{E}\left[\big\|S_T^0\big\|_2\right] + \sum_{i=1}^d \mathbb{E}\left[\big\|S_T^{e_i}\big\|_2\right]$$

and that

$$\big\|\mathbb{E}\big[\,\|\mathcal{A}_{\mathfrak{M},\mathfrak{N}}\|_2^\nu\,\big]\big\|_{\mathcal{L}^2(\mathbb{P}_{X_d})} \le \Big(\int_{[u,v]^d}\big[\mathfrak{c}_\nu(\|x\|_2)\big]^2\,d\mathbb{P}_{X_d}(x)\Big)^{1/2} \le \mathfrak{c}_\nu(\sqrt{d}\max\{1,|u|,|v|\})$$

this implies the desired estimates. ∎

In the next lemma we show that the average of the composition of a neural network with different affine functions can be represented by a single neural network and we bound the number and size of its parameters.

[16] Recall that for a matrix $M \in \mathbb{R}^{d\times d}$ we denote by $\|M\|_2 := \big(\sum_{i,j=1}^d M_{ij}^2\big)^{1/2}$ its Frobenius norm.

Lemma A.5 (compositions of neural networks and affine functions). *Assume Setting 2.5. Let $d, n \in \mathbb{N}$, $\mathbf{b} \in \mathbf{A}_d$, $\boldsymbol{\eta} \in \mathcal{P}_{\mathbf{b}}$, and*

$$((M^{(j)}, N^{(j)}))_{j=1}^{n} \in \big(\mathbb{R}^{d\times d} \times \mathbb{R}^d\big)^n.$$

Then there exist $\mathbf{a} \in \mathbf{A}_d$ and $\boldsymbol{\theta} \in \mathcal{P}_{\mathbf{a}}$ such that it holds that

(i) $\mathcal{F}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{j=1}^{n} \mathcal{F}(\boldsymbol{\eta}) \circ \mathcal{A}_{M^{(j)},N^{(j)}}$,

(ii) $P(\mathbf{a}) \leq n^2 P(\mathbf{b})$,

(iii) $\|\boldsymbol{\theta}\|_\infty \leq \sqrt{d}\|\boldsymbol{\eta}\|_\infty \max_{j=1}^{n}\big(\|M^{(j)}\|_2 + \|N^{(j)}\|_2 + 1\big)$,

(iv) $L(\mathbf{a}) = L(\mathbf{b})$, *and*

(v) $\|\mathbf{a}\|_\infty = n\|\mathbf{b}\|_\infty$.

Proof of Lemma A.5. With the exception of item (iii) this result is proven in [29, Lemma 3.8]. There it is shown that for $\boldsymbol{\eta} = ((V_l, A_l))_{l=1}^{L}$ a suitable parametrization $\boldsymbol{\theta} = ((W_l, B_l))_{l=1}^{L}$ is given by $W_L := \begin{bmatrix}\frac{1}{n}V_L & \frac{1}{n}V_L & \dots & \frac{1}{n}V_L\end{bmatrix}$, $B_L := A_L$,

$$W_1 := \begin{bmatrix} V_1M^{(1)} \\ \vdots \\ V_1M^{(n)} \end{bmatrix}, \quad B_1 := \begin{bmatrix} V_1N^{(1)} + A_1 \\ \vdots \\ V_1N^{(n)} + A_1 \end{bmatrix}, \quad \text{and} \quad W_l := \begin{bmatrix} V_l & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & V_l \end{bmatrix}, \quad B_l := \begin{bmatrix} A_l \\ \vdots \\ A_l \end{bmatrix},$$

$l \in \{2, \dots, L-1\}$. Now observe that

$$\|W_1\|_\infty \leq \sqrt{d}\|\boldsymbol{\eta}\|_\infty \max_{j=1}^{n}\|M^{(j)}\|_2 \quad \text{and} \quad \|B_1\|_\infty \leq \sqrt{d}\|\boldsymbol{\eta}\|_\infty \max_{j=1}^{n}\big(\|N^{(j)}\|_2 + 1\big),$$

which proves the lemma. ∎

Now we are ready to prove Theorem 3.3.

Proof of Theorem 3.3. Fix $d \in \mathbb{N}$, $\varepsilon \in (0,1)$ and define $\mathfrak{m} := \max\{1, |u|, |v|\}$. Let $\mathfrak{M}$, $\mathfrak{N}$, $\mathfrak{c}_1(1)$, and $\mathfrak{c}_\nu(\mathfrak{m})$ be given as in Lemma A.4, let $((\mathfrak{M}^{(j)}, \mathfrak{N}^{(j)}))_{j\in\mathbb{N}}$ be i.i.d. random variables with $(\mathfrak{M}^{(1)}, \mathfrak{N}^{(1)}) \sim (\mathfrak{M}, \mathfrak{N})$, and let

$$n \in \big[16D^2\varepsilon^{-1}, 32D^2\varepsilon^{-1}\big) \cap \mathbb{N} \quad \text{and} \quad \delta := \big(8\mathfrak{c}_\nu(\mathfrak{m})\big)^{-1} d^{-\nu/2}\varepsilon^{1/2}. \tag{A.9}$$

Define $g := \mathcal{F}(\boldsymbol{\eta}_{d,\delta})$ and note that Setting 3.1, Lemma A.4, and the Feynman–Kac formula [29, Corollary 2.23(ii)] establish that for every $x \in \mathbb{R}^d$ it holds that

$$|g(x)| \leq D, \quad |\varphi_d(x) - g(x)| \leq \delta(1 + \|x\|_2^\nu), \quad \text{and} \quad F_d(T,x) = \mathbb{E}\left[\big(\varphi_d \circ \mathcal{A}_{\mathfrak{M},\mathfrak{N}}\big)(x)\right]. \tag{A.10}$$

We now use techniques from [29, Proof of Proposition 3.4] to show that the random variable $G\colon \Omega \to [0,\infty)$, given by

$$G := \varepsilon^{-1/2} \underbrace{\Big\| F_d(T,\cdot) - \tfrac{1}{n}\sum_{j=1}^{n} g \circ \mathcal{A}_{\mathfrak{M}^{(j)},\mathfrak{N}^{(j)}} \Big\|_{\mathcal{L}^2(\mathbb{P}_{X_d})}}_{:=G_1} + \big(6\mathfrak{c}_1(1)dn\big)^{-1} \underbrace{\max_{j=1}^{n}\Big(\big\|\mathfrak{M}^{(j)}\big\|_2 + \big\|\mathfrak{N}^{(j)}\big\|_2\Big)}_{:=G_2},$$

satisfies $\mathbb{E}[G] \leq 1$. First, note that (A.9) and (A.10), together with Jensen's inequality, Fubini's theorem, the Bienaymé formula (see also [29, Lemma 2.3]), and Lemma A.4, ensure that

$$\begin{aligned}
\mathbb{E}[G_1] &\leq \left\| \mathbb{E}\left[(\varphi_d - g) \circ \mathcal{A}_{\mathfrak{M},\mathfrak{N}}\right] \right\|_{\mathcal{L}^2(\mathbb{P}_{X_d})} + \mathbb{E}\left[\left\| \mathbb{E}\left[g \circ \mathcal{A}_{\mathfrak{M},\mathfrak{N}}\right] - \tfrac{1}{n}\sum_{j=1}^{n} g \circ \mathcal{A}_{\mathfrak{M}^{(j)},\mathfrak{N}^{(j)}} \right\|_{\mathcal{L}^2(\mathbb{P}_{X_d})} \right] \\
&\leq \left\| \mathbb{E}\big[\delta(1 + \|\mathcal{A}_{\mathfrak{M},\mathfrak{N}}\|_2^{\nu})\big] \right\|_{\mathcal{L}^2(\mathbb{P}_{X_d})} \\
&\quad + \Big(\mathbb{E}\Big[\int_{[u,v]^d} \Big(\mathbb{E}\left[(g \circ \mathcal{A}_{\mathfrak{M},\mathfrak{N}})(x)\right] - \tfrac{1}{n}\sum_{j=1}^{n} \big(g \circ \mathcal{A}_{\mathfrak{M}^{(j)},\mathfrak{N}^{(j)}}\big)(x) \Big)^2 d\mathbb{P}_{X_d}(x) \Big] \Big)^{1/2} \\
&\leq \delta\big(1 + \big\| \mathbb{E}\big[\|\mathcal{A}_{\mathfrak{M},\mathfrak{N}}\|_2^{\nu} \big] \big\|_{\mathcal{L}^2(\mathbb{P}_{X_d})}\big) + \Big(\int_{[u,v]^d} \mathbb{V}\Big[\tfrac{1}{n}\sum_{j=1}^{n} \big(g \circ \mathcal{A}_{\mathfrak{M}^{(j)},\mathfrak{N}^{(j)}}\big)(x) \Big] d\mathbb{P}_{X_d}(x) \Big)^{1/2} \\
&\leq 2\mathfrak{c}_{\nu}(\mathfrak{m}) d^{\nu/2}\delta + Dn^{-1/2} \leq \tfrac{1}{2}\varepsilon^{1/2}.
\end{aligned}$$

Next, observe that Lemma A.4 establishes that

$$\mathbb{E}[G_2] \leq \mathbb{E}\Big[\sum_{j=1}^{n} \Big(\|\mathfrak{M}^{(j)}\|_2 + \|\mathfrak{N}^{(j)}\|_2 \Big) \Big] \leq n\mathbb{E}\big[\|\mathfrak{M}\|_2 + \|\mathfrak{N}\|_2 \big] \leq 3\mathfrak{c}_1(1) dn$$

which proves that $\mathbb{E}[G] = \varepsilon^{-1/2}\mathbb{E}[G_1] + \big(6\mathfrak{c}_1(1)dn\big)^{-1}\mathbb{E}[G_2] \leq 1$. Thus, there exists $\omega \in \Omega$ such that $G(\omega) \leq 1$ (see [29, Proposition 3.3]), and with

$$M^{(j)} := \mathfrak{M}^{(j)}(\omega) \quad \text{and} \quad N^{(j)} := \mathfrak{N}^{(j)}(\omega), \quad j \in \{1, \ldots, n\},$$

it holds that

$$\tfrac{1}{(v-u)^d} \Big\| F_d(T, \cdot) - \tfrac{1}{n}\sum_{j=1}^{n} g \circ \mathcal{A}_{M^{(j)},N^{(j)}} \Big\|^2_{\mathcal{L}^2([u,v]^d)} = G_1^2(\omega) \leq G^2(\omega)\varepsilon \leq \varepsilon \tag{A.11}$$

and that

$$\max_{j=1}^{n} \Big(\|M^{(j)}\|_2 + \|N^{(j)}\|_2 \Big) = G_2(\omega) \leq 6\mathfrak{c}_1(1) G(\omega) dn \leq 192 D^2 \mathfrak{c}_1(1) d\varepsilon^{-1}.$$

By Lemma A.5, our assumptions, and (A.9) there exist $\mathbf{a} \in \mathbf{A}_d$ and $\boldsymbol{\theta} \in \mathcal{P}_{\mathbf{a}}$ satisfying the following:

(i) $\operatorname{clip}_D \circ \mathcal{F}(\boldsymbol{\theta}) = \mathcal{F}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{j=1}^{n} \mathcal{F}(\boldsymbol{\eta}_{d,\delta}) \circ \mathcal{A}_{M^{(j)},N^{(j)}} = \frac{1}{n}\sum_{j=1}^{n} g \circ \mathcal{A}_{M^{(j)},N^{(j)}}$;

(ii) $P(\mathbf{a}) \leq n^2 P(\mathbf{b}_{d,\delta}) \leq 32^2 D^4 \zeta d^{\gamma} \varepsilon^{-2} \delta^{-\lambda} \leq C d^{\nu\lambda/2+\gamma} \varepsilon^{-\lambda/2-2}$;

(iii) $\|\boldsymbol{\theta}\|_{\infty} \leq \sqrt{d}\|\boldsymbol{\eta}_{d,\delta}\|_{\infty}\big(192 D^2 \mathfrak{c}_1(1) d\varepsilon^{-1} + 1\big) \leq C d^{(\nu\kappa+3)/2+\beta} \epsilon^{-\kappa/2-1}$;

(iv) $L(\mathbf{a}) = L(\mathbf{b}_{d,\delta}) = L(\mathbf{b}_{d,cd^{-\nu/2}\varepsilon^{1/2}})$; and

(v) $\|\mathbf{a}\|_{\infty} = n\|\mathbf{b}_{d,\delta}\|_{\infty} \leq 32 D^2 \varepsilon^{-1} \|\mathbf{b}_{d,\delta}\|_{\infty} \leq C\varepsilon^{-1} \|\mathbf{b}_{d,cd^{-\nu/2}\varepsilon^{1/2}}\|_{\infty}$,

where $C := \zeta \max\{32^2 D^4 (8\mathfrak{c}_\nu(\mathfrak{m}))^\lambda, (192 D^2 \mathfrak{c}_1(1) + 1)(8\mathfrak{c}_\nu(\mathfrak{m}))^\kappa\}$ and $c := (8\mathfrak{c}_\nu(\mathfrak{m}))^{-1}$. Together with (A.11) this proves the theorem. ∎

Acknowledgements. The authors are grateful to Shahar Mendelson and Stefan Steinerberger for their useful comments.

REFERENCES

[1] C. Aliprantis and K. Border, *Infinite Dimensional Analysis: A Hitchhiker's Guide (third edition)*, Springer, 2007.
[2] Z. Allen-Zhu, Y. Li, and Z. Song, *A convergence theory for deep learning via over-parameterization*, in International Conference on Machine Learning, 2019, pp. 242–252.
[3] W. Ames, *Numerical Methods for Partial Differential Equations*, Comput. Sci. Sci. Comput., Elsevier Science, 2014.
[4] M. Anthony and P. Bartlett, *Neural Network Learning: Theoretical Foundations*, Cambridge University Press, 2009.
[5] L. Arnold, *Stochastic differential equations*, A Wiley-Interscience publication, Wiley, 1974.
[6] S. Arora, R. Ge, B. Neyshabur, and Y. Zhang, *Stronger generalization bounds for deep nets via a compression approach*, in International Conference on Machine Learning, 2018, pp. 254–263.
[7] P. L. Bartlett, O. Bousquet, S. Mendelson, et al., *Local rademacher complexities*, The Annals of Statistics, 33 (2005), pp. 1497–1537.
[8] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, *Spectrally-normalized margin bounds for neural networks*, in Advances in Neural Information Processing Systems, 2017, pp. 6240–6249.
[9] P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian, *Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks.*, J. Mach. Learn. Res., 20 (2019), pp. 63–1.
[10] C. Beck, S. Becker, P. Grohs, N. Jaafari, and A. Jentzen, *Solving stochastic differential equations and Kolmogorov equations by means of deep learning*, arXiv:1806.00421, (2018).
[11] C. Beck, W. E, and A. Jentzen, *Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations*, Journal of Nonlinear Science, 29 (2019), pp. 1563–1619.
[12] S. Becker, P. Cheridito, and A. Jentzen, *Deep optimal stopping*, Journal of Machine Learning Research, 20 (2019), pp. 1–25.
[13] J. Berner, D. Elbrächter, and P. Grohs, *How degenerate is the parametrization of neural networks with the ReLU activation function?*, in Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 7790–7801.
[14] J. Berner, D. Elbrächter, P. Grohs, and A. Jentzen, *Towards a regularity theory for ReLU networks – chain rule and global error estimates*, in 2019 13th International conference on Sampling Theory and Applications (SampTA), 2019, pp. 1–5.
[15] H. Bölcskei, P. Grohs, G. Kutyniok, and P. Petersen, *Optimal approximation with sparsely connected deep neural networks*, SIAM Journal on Mathematics of Data Science, 1 (2019), pp. 8–45.
[16] M. Burger and A. Neubauer, *Error bounds for approximation with neural networks*, Journal of Approximation Theory, 112 (2001), pp. 235–250.
[17] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, *The loss surfaces of multilayer networks*, in Artificial Intelligence and Statistics, 2015, pp. 192–204.
[18] F. Cucker and S. Smale, *On the mathematical foundations of learning*, Bulletin of the American mathematical society, 39 (2002), pp. 1–49.
[19] F. Cucker and D. X. Zhou, *Learning Theory: An Approximation Theory Viewpoint*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2007.
[20] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, *Gradient descent finds global minima of deep neural networks*, in International Conference on Machine Learning, 2019, pp. 1675–1685.
[21] W. E, J. Han, and A. Jentzen, *Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations*, Communications in Math-

ematics and Statistics, 5 (2017), pp. 349–380.
[22] W. E and B. Yu, *The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems*, Communications in Mathematics and Statistics, 6 (2018), pp. 1–12.
[23] D. Elbrächter, P. Grohs, A. Jentzen, and C. Schwab, *DNN expression rate analysis of high-dimensional PDEs: Application to option pricing*, arXiv:1809.07669, (2018).
[24] L. Evans, *Partial Differential Equations (second edition)*, Graduate studies in mathematics, American Mathematical Society, 2010.
[25] M. Fujii, A. Takahashi, and M. Takahashi, *Asymptotic expansion as prior knowledge in deep learning method for high dimensional BSDEs*, Asia-Pacific Financial Markets, 26 (2019), pp. 391–408.
[26] K.-I. Funahashi, *On the approximate realization of continuous mappings by neural networks*, Neural Networks, 2 (1989), pp. 183–192.
[27] N. Golowich, A. Rakhlin, and O. Shamir, *Size-independent sample complexity of neural networks*, in Conference On Learning Theory, 2018, pp. 297–299.
[28] C. Graham and D. Talay, *Stochastic Simulation and Monte Carlo Methods: Mathematical Foundations of Stochastic Simulation*, Stochastic Modelling and Applied Probability, Springer Berlin Heidelberg, 2013.
[29] P. Grohs, F. Hornung, A. Jentzen, and P. von Wurstemberger, *A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of black-scholes partial differential equations*, arXiv:1809.02362v1, (2018). To appear in Mem. Amer. Math. Soc.
[30] P. Grohs, F. Hornung, A. Jentzen, and P. Zimmermann, *Space-time error estimates for deep neural network approximations for differential equations*, arXiv:1908.03833, (2019).
[31] L. Györfi, M. Kohler, A. Krzyzak, and H. Walk, *A distribution-free theory of nonparametric regression*, Springer Science & Business Media, 2006.
[32] M. Hairer, M. Hutzenthaler, and A. Jentzen, *Loss of regularity for Kolmogorov equations*, Ann. Probab., 43 (2015), pp. 468–527.
[33] J. Han, A. Jentzen, and W. E, *Solving high-dimensional partial differential equations using deep learning*, Proceedings of the National Academy of Sciences, 115 (2018), pp. 8505–8510.
[34] P. Henry-Labordere, *Deep primal-dual algorithm for BSDEs: Applications of machine learning to cva and im*, SSRN Electronic Journal, (2017).
[35] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, *Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups*, IEEE Signal Processing Magazine, 29 (2012), pp. 82–97.
[36] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, Journal of the American Statistical Association, 58 (1963), pp. 13–30.
[37] M. Hutzenthaler, A. Jentzen, T. Kruse, and T. A. Nguyen, *A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations*, SN Partial Differential Equations and Applications, 1 (2020), pp. 1–34.
[38] A. Jentzen, D. Salimova, and T. Welti, *A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients*, arXiv:1809.07321, (2018).
[39] K. Kawaguchi, *Deep learning without poor local minima*, in Advances in neural information processing systems, 2016, pp. 586–594.
[40] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv:1412.6980, (2014).
[41] V. Koltchinskii, *Introduction*, in Oracle Inequalities in Empirical Risk Minimization and Sparse Recovery Problems, Springer, 2011, pp. 1–16.
[42] G. Kutyniok, P. Petersen, M. Raslan, and R. Schneider, *A theoretical analysis of deep neural networks and parametric PDEs*, arXiv:1904.00377, (2019).
[43] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, nature, 521 (2015), p. 436.
[44] Y. LeCun, C. Cortes, and C. J. C. Burges, *The MNIST database of handwritten digits*, 1998. http://yann.lecun.com/exdb/mnist/ [online; accessed August 22, 2018].
[45] Y. Li and Y. Liang, *Learning overparameterized neural networks via stochastic gradient descent on structured data*, in Advances in Neural Information Processing Systems, 2018, pp. 8157–8166.
[46] Y. Li and Y. Yuan, *Convergence analysis of two-layer neural networks with ReLU activation*, in Advances

in Neural Information Processing Systems, 2017, pp. 597–607.
[47] P. Massart, *Concentration inequalities and model selection*, Springer, 2007.
[48] S. Mei, A. Montanari, and P.-M. Nguyen, *A mean field view of the landscape of two-layer neural networks*, Proceedings of the National Academy of Sciences, 115 (2018), pp. E7665–E7671.
[49] M. Mohri, A. Rostamizadeh, A. Talwalkar, and F. Bach, *Foundations of Machine Learning*, Adaptive computation and machine learning series, MIT Press, 2012.
[50] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, *Exploring generalization in deep learning*, in Advances in Neural Information Processing Systems, 2017, pp. 5947–5956.
[51] M. Nielsen, *Neural networks and deep learning*, 2015. http://neuralnetworksanddeeplearning.com/chap1.html [online; accessed March 05, 2018].
[52] D. Perekrestenko, P. Grohs, D. Elbrächter, and H. Bölcskei, *The universal approximation power of finite-width deep ReLU networks*, arXiv:1806.01528, (2018).
[53] P. Petersen, M. Raslan, and F. Voigtlaender, *Topological properties of the set of functions generated by neural networks of fixed size*, arXiv:1806.08459, (2018).
[54] P. Petersen and F. Voigtlaender, *Optimal approximation of piecewise smooth functions using deep ReLU neural networks*, Neural Networks, 108 (2018), pp. 296–330.
[55] C. Reisinger and Y. Zhang, *Rectified deep neural networks overcome the curse of dimensionality for nonsmooth value functions in zero-sum games of nonlinear stiff systems*, arXiv:1903.06652, (2019).
[56] C. Schwab and J. Zech, *Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in uq*, Analysis and Applications, 17 (2019), pp. 19–55.
[57] R. Seydel, *Tools for Computational Finance*, Universitext, Springer London, 2012.
[58] U. Shaham, A. Cloninger, and R. R. Coifman, *Provable approximation properties for deep neural networks*, Applied and Computational Harmonic Analysis, 44 (2018), pp. 537 – 557.
[59] O. Shamir and T. Zhang, *Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes*, in International Conference on Machine Learning, 2013, pp. 71–79.
[60] J. Sirignano and K. Spiliopoulos, *DGM: A deep learning algorithm for solving partial differential equations*, Journal of Computational Physics, 375 (2018), pp. 1339–1364.
[61] S. A. Van de Geer, *Applications of empirical process theory, volume 6 of cambridge series in statistical and probabilistic mathematics*, 2000.
[62] D. Yarotsky, *Error bounds for approximations with deep ReLU networks*, Neural Networks, 94 (2017), pp. 103–114.
[63] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, *Understanding deep learning requires rethinking generalization*, in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, 2017.

IV Numerically Solving Parametric Families of High-Dimensional Kolmogorov Partial Differential Equations via Deep Learning

Comments

Conference: Poster presentation at NeurIPS 2020.
Code: `github.com/juliusberner/deep_kolmogorov`
E-Print: `arXiv:2011.04602[cs.LG]`

Contribution

Developed and written by Markus Dablander and Julius Berner in equal parts. Philipp Grohs contributed initial ideas and scientific advice. The numerical experiments have been implemented and conducted by Julius Berner.

Bibliographic Information

Berner, Julius, Markus Dablander, and Philipp Grohs (2020). "Numerically Solving Parametric Families of High-Dimensional Kolmogorov Partial Differential Equations via Deep Learning." In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 16615–16627.

Numerically Solving Parametric Families of High-Dimensional Kolmogorov Partial Differential Equations via Deep Learning

Julius Berner[*]
Faculty of Mathematics, University of Vienna
Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
julius.berner@univie.ac.at

Markus Dablander[*]
Mathematical Institute, University of Oxford
Andrew Wiles Building, OX2 6GG, Oxford, United Kingdom
markus.dablander@maths.ox.ac.uk

Philipp Grohs
Faculty of Mathematics and Research Platform DataScience@UniVienna, University of Vienna
Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
RICAM, Austrian Academy of Sciences
Altenberger Straße 69, 4040 Linz, Austria
philipp.grohs@univie.ac.at

Abstract

We present a deep learning algorithm for the numerical solution of parametric families of high-dimensional linear Kolmogorov partial differential equations (PDEs). Our method is based on reformulating the numerical approximation of a whole family of Kolmogorov PDEs as a single statistical learning problem using the Feynman-Kac formula. Successful numerical experiments are presented, which empirically confirm the functionality and efficiency of our proposed algorithm in the case of heat equations and Black-Scholes option pricing models parametrized by affine-linear coefficient functions. We show that a single deep neural network trained on simulated data is capable of learning the solution functions of an entire family of PDEs on a full space-time region. Most notably, our numerical observations and theoretical results also demonstrate that the proposed method does not suffer from the curse of dimensionality, distinguishing it from almost all standard numerical methods for PDEs.

1 Introduction

Linear parabolic partial differential equations (PDEs) of the form

$$\frac{\partial u_\gamma}{\partial t} = \frac{1}{2}\,\mathrm{Trace}\left(\sigma_\gamma[\sigma_\gamma]^*\nabla_x^2 u_\gamma\right) + \langle \mu_\gamma, \nabla_x u_\gamma\rangle, \quad u_\gamma(x,0) = \varphi_\gamma(x), \tag{1}$$

are referred to as Kolmogorov PDEs, see [23] for a thorough study of their mathematical properties. Throughout this paper, the functions

$$\varphi_\gamma : \mathbb{R}^d \to \mathbb{R} \quad \text{(initial condition)} \quad \text{and} \quad \sigma_\gamma : \mathbb{R}^d \to \mathbb{R}^{d\times d}, \quad \mu_\gamma : \mathbb{R}^d \to \mathbb{R}^d \quad \text{(coefficient maps)}$$

[*]Equal contribution.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

are continuous, and are implicitly determined by a real parameter vector $\gamma \in D$, whereby D is a compact set in Euclidean space.

Kolmogorov PDEs frequently appear in applications in a broad variety of scientific disciplines, including physics and financial engineering [9, 41, 56]. In particular, note that the heat equation from physical modelling as well as the widely-known Black-Scholes equation from computational finance are important special cases of Equation (1). Typically, one is interested in finding the (viscosity) solution[2]

$$u_\gamma : [v, w]^d \times [0, T] \to \mathbb{R}$$

of Equation (1) on a predefined space-time region of the form $[v, w]^d \times [0, T]$. In almost all cases, however, Kolmogorov PDEs cannot be solved explicitly. Furthermore, standard numerical solution algorithms for PDEs, in particular those based on a discretization of the considered domain, are known to suffer from the so-called *curse of dimensionality*[3], meaning that their computational cost grows exponentially in the dimension of the domain [2, 50]. The development of new, computationally efficient methods for the numerical solution of Kolmogorov PDEs is therefore of high interest for applied scientists.

In this work, we present a novel deep learning algorithm capable of numerically approximating the solutions $(u_\gamma)_{\gamma \in D}$ of a whole family of γ-parametrized Kolmogorov PDEs on a full space-time region. Specifically, our proposed method allows to train a single deep neural network

$$\Phi \colon D \times [v, w]^d \times [0, T] \to \mathbb{R} \tag{2}$$

to approximate the *parametric solution map*

$$\bar{u} : D \times [v, w]^d \times [0, T] \to \mathbb{R}, \quad (\gamma, x, t) \mapsto \bar{u}(\gamma, x, t) := u_\gamma(x, t), \tag{3}$$

of a family of γ-parametrized Kolmogorov PDEs on the generalized domain $D \times [v, w]^d \times [0, T]$. Most notably, we also theoretically investigate the associated approximation and generalization errors and rigorously show that our algorithm does not suffer from the curse of dimensionality with respect to the neural network size as well as the sample size. We emphasize that our empirical results strongly suggest that also the empirical risk minimization (ERM) algorithm, usually a variant of stochastic gradient descent, does not suffer from the curse of dimensionality but proving this is out of scope of this paper.

1.1 PDEs and Deep Learning: Current Research and Related Work

Interest in deep-learning based techniques for the numerical solution of PDEs has been growing rapidly in recent years [6, 24, 32, 46, 51, 54, 55]. This sharp rise in interest can partly be explained by the remarkable ability of deep neural networks to avoid incurring the curse of dimensionality when used to approximate PDE solutions in particular settings. More specifically, in some situations it has been possible to find theoretical upper bounds for the size of the required neural network architectures which do not depend exponentially on the dimension of the PDE [14, 20, 29, 49, 47]. This represents a rare and crucial advantage over classical finite difference and finite element methods, all of which typically cannot be used in high dimensions due to the resulting exponential explosion of required computational costs.

As a result of these successes, deep learning has recently been studied as a numerical solution technique for the more general group of parametric PDEs and their associated parametric solution maps [12, 27, 33, 35, 36, 49]. The investigation of the capabilities of deep artificial neural networks to learn parametric solution maps of specific parametrizable families of PDEs has become a new and active area of research. In this work, we provide novel theoretical and empirical results which, for the first time, demonstrate the viability of deep learning algorithms for the scalable solution of large classes of parametric Kolmogorov PDEs.

The formulation of the learning problem underlying our method is inspired by the work of Beck et al. [5]. There it is shown how deep neural networks can be used to numerically solve a non-parametric version of Equation (1) with fixed initial condition φ_γ and fixed coefficients maps $\sigma_\gamma, \mu_\gamma$

[2] Viscosity solutions are the appropriate solution concept for a wide range of PDEs [10, 23]. Viscosity solutions are continuous, but not necessarily differentiable.

[3] The classical way to circumvent the curse of dimensionality has been the employment of stochastic Monte Carlo based methods, see e.g. [19]; these techniques, however, are only suitable to approximately compute the solution $u_\gamma(x, t)$ at a single *fixed* space-time point $(x, t) \in [v, w]^d \times [0, T]$, limiting their usefulness in practice.

on a predefined space region $[v, w]^d$ and at a predefined time slice $T > 0$. In other words, their non-parametric method allows to approximate the function

$$u_\gamma(\,\cdot\,, T) : [v, w]^d \to \mathbb{R}, \quad x \mapsto u_\gamma(x, T),$$

for fixed $\gamma \in D$ by training a deep neural network with suitable simulated data of the form

$$(X, \varphi_\gamma(S_{\gamma,X,T})) \in [v, w]^d \times \mathbb{R}.$$

Here, X is uniformly drawn from the spatial hypercube $[v, w]^d$ and the random vector $S_{\gamma,X,T}$ is the value of the solution process $(S_{\gamma,X,t})_{t\geq 0}$ of the stochastic differential equation (SDE)

$$dS_{\gamma,X,t} = \mu_\gamma(S_{\gamma,X,t})dt + \sigma_\gamma(S_{\gamma,X,t})dB_t, \quad S_{\gamma,X,0} = X,$$

at time $t = T$, whereby $(B_t)_{t\geq 0}$ is a standard d-dimensional Brownian motion.

The choice of training data is based on the following important identity, which under suitable regularity assumptions holds for all $x \in [v, w]^d$, $t \in [0, T]$, and $\gamma \in D$:

$$u_\gamma(x, t) = \mathbb{E}[\varphi_\gamma(S_{\gamma,x,t})]. \tag{4}$$

Equality (4) is a version of the well-known *Feynman-Kac formula* which establishes a link between the theory of parabolic PDEs and the theory of stochastic differential equations [23]. Using the Feynman-Kac formula, one can show within the mathematical framework of empirical risk minimization [11, 53] that $u_\gamma(\cdot\,, T)$ is in fact the solution of the supervised statistical learning problem defined by the predictor variable X, the target variable $\varphi_\gamma(S_{\gamma,X,T})$, and a standard quadratic loss function [5].

1.2 Novel Contribution

In this work, we introduce the first algorithm for the numerical solution of *parametric* Kolmogorov PDEs on a *whole* space-time region. No previous technique has achieved this degree of generality; all former methods for parametric Kolmogorov PDEs were either only capable of computing local solutions at single space-time points of the domain using Monte Carlo based approaches or did not employ deep neural networks and were thus not able to break the curse of dimensionality. Our technique is made possible by constructing a suitable supervised learning problem via a nontrivial application of the Feynman-Kac formula (4), which involves random initial conditions and SDEs with random coefficients and stopping times. This reformulation of a broad class of parametric PDEs as learning problems provides a new theoretical framework to analyze the convergence behavior of deep learning algorithms. Building upon this framework, we prove theoretical guarantees for the computational performance of our technique and, to the best of our knowledge, establish the first combined approximation and generalization results for parametric PDEs.

Note that the parametric nature of the presented algorithm opens up the novel possibility to study changes in the potentially high-dimensional solution manifold of Equation (1) as its initial conditions and coefficient maps vary with $\gamma \in D$. The study of such changes is commonly referred to as *sensitivity analysis*. Recall that the proposed method delivers a neural network Φ which approximates the parametric PDE solution map, i.e. $\Phi \approx \bar{u}$. The partial derivatives of Φ with respect to the parameter γ, the spatial variable x, and the time variable t can then be readily computed via automatic differentiation. Thus, the partial derivatives of Φ become computationally accessible approximations of the partial derivatives of $\bar{u}$. The partial derivatives of $\bar{u}$ in turn play an important role in a variety of widespread applications, such as in the computation of the "Greeks" associated with the Black-Scholes model from financial engineering (see Section 3.1).

Another highly relevant application area opened up by our method is the calibration of the usually unknown PDE coefficients $\sigma_\gamma, \mu_\gamma$ using real-world data. After solving a parametric PDE with our technique, one can fit γ such that the PDE solution manifold best describes a real data set and additionally apply uncertainty quantification techniques if γ is modelled as a random variable.

Finally, we establish a new architecture and compare different learning schemes to provide future researchers with a robust framework for parametric PDEs, which are inherently less stable than their simpler non-parametric counterparts. Further, this work is complemented by an extendable implementation with the possibility of distributed training and hyperparameter optimization for the special use-cases of other researchers.

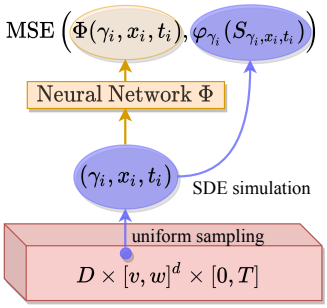


Figure 1: Illustration of the proposed supervised learning problem with predictor variable Λ and target variable $\varphi_\Gamma(S_{\Gamma,X,\mathcal{T}})$.

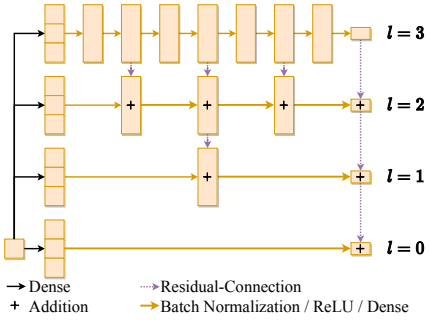


Figure 2: Illustration of the Multilevel architecture for $L = 4$, $q = 3$.

2 Algorithm

The key idea of the presented algorithm is to describe the parametric solution map (3) of the γ-parametrized Kolmogorov PDE (1) as the regression function of an appropriately chosen supervised statistical learning problem. One can then use simulated training data in order to learn $\bar{u}$ by means of deep learning. Inspired by the Feynman-Kac formula (4), we construct a new statistical learning problem via a uniformly distributed predictor variable and a statistically dependent target variable:

$$\Lambda := (\Gamma, X, \mathcal{T}) \in D \times [v, w]^d \times [0, T] \quad \text{(predictor)} \quad \text{and} \quad Y := \varphi_\Gamma(S_\Lambda) \in \mathbb{R} \quad \text{(target)}.$$

The random vector S_Λ is defined as the value of the solution process $(S_{\Gamma,X,t})_{t\geq 0}$ of the Γ-parametrized stochastic differential equation

$$dS_{\Gamma,X,t} = \mu_\Gamma(S_{\Gamma,X,t})dt + \sigma_\Gamma(S_{\Gamma,X,t})dB_t, \quad S_{\Gamma,X,0} = X, \tag{5}$$

at the (random) stopping time $t = \mathcal{T}$. For suitable regularity assumptions, the Feynman-Kac formula (4) then ensures that

$$\mathbb{E}[\,Y \mid \Lambda = (\gamma, x, t)] = \mathbb{E}[\varphi_\Gamma(S_\Lambda) \mid \Lambda = (\gamma, x, t)] = \mathbb{E}[\varphi_\gamma(S_{\gamma,x,t})] = u_\gamma(x, t) = \bar{u}(\gamma, x, t).$$

This shows that the minimizer of the corresponding statistical learning problem with quadratic loss function is indeed the parametric Kolmogorov PDE solution map, see Theorem A.1 in the appendix for the precise assumptions and a detailed proof.

Theorem 1 (Learning Problem)**.** *It holds that the parametric solution map $\bar{u}$ is the unique minimizer of the statistical learning problem*

$$\min_f \mathbb{E}\big[\big(f(\Lambda) - Y\big)^2\big]. \tag{6}$$

Restricting to a hypothesis space of suitable neural networks $\mathcal{H}$ and minimizing the empirical mean squared error (MSE) loss corresponding to (6), we arrive at the feasible supervised ERM problem

$$\min_{\Phi\in\mathcal{H}} \tfrac{1}{s}\textstyle\sum_{i=1}^s (\Phi(\lambda_i) - y_i)^2 \tag{7}$$

where $((\lambda_i, y_i))_{i=1}^s$ are realizations of i.i.d. samples drawn from the distribution of (Λ, Y). Typically, this problem is then solved by a variant of stochastic gradient descent [48]. The algorithm is graphically illustrated in Figure 1.

It is trivial to simulate i.i.d. samples of the predictor variable Λ, due to its uniform distribution. On the other hand i.i.d. samples of the target variable $Y = \varphi_\Gamma(S_\Lambda)$ can be obtained via standard numerical SDE solution techniques without curse of dimensionality [34]. An example for such a technique is given by the *Euler-Maruyama approximation* with $M \in \mathbb{N}$ equidistant steps $(S_\Lambda^{M,m})_{m=0}^M$ which is defined by the following scheme:

$$S_\Lambda^{M,0} = X \quad \text{and} \quad S_\Lambda^{M,m+1} = S_\Lambda^{M,m} + \mu_\Gamma(S_\Lambda^{M,m})\tfrac{\mathcal{T}}{M} + \sigma_\Gamma(S_\Lambda^{M,m})\big(B_{\frac{(m+1)\mathcal{T}}{M}} - B_{\frac{m\mathcal{T}}{M}}\big). \tag{8}$$

The following theorem shows that solving the learning problem with data simulated by the Euler-Maruyama scheme does indeed result in the expected approximation of the parametric PDE solution map $\bar{u}$, see Theorem A.2 in the appendix for a proof.

Theorem 2 (Approximated Learning Problem)**.** *The unique minimizer $\bar{u}^M$ of the approximated statistical learning problem*

$$\min_f \mathbb{E}\big[\big(f(\Lambda) - Y^M\big)^2\big]$$

where $Y^M := \varphi_\Gamma(S_\Lambda^{M,M})$ is simulated using the Euler-Maruyama scheme (8) *with $M \sim 1/\varepsilon^2$ equidistant steps satisfies that*

$$\|\bar{u}^M - \bar{u}\|_{\mathcal{L}^\infty(D\times[v,w]^d\times[0,T])} \leq \varepsilon.$$

In other words, the approximation of the SDE solution $S_\Lambda^{M,M} \approx S_\Lambda$ carries over to the approximation of the corresponding minimizer $\bar{u}^M \approx \bar{u}$. Therefore there exists no constraint of having to solve the SDE in (5) analytically. The ability to easily simulate artificial training data opens up the highly desirable capability to supply the learning algorithm with a potentially infinite stream of i.i.d. data samples. Instead of having to use a train/val/test split on a given finite data set, one can thus constantly simulate new data points on demand during training. Since the number of samples then grows at will parallel to the training process, the first epoch never finishes and every new gradient computation can be done on the basis of previously unseen data.

2.1 Example: Affine-Linear Coefficient Functions

In Section 3 we present numerical experiments based on our algorithm for the important special case where σ_γ and μ_γ are affine-linear functions. Thus from now on let us assume that[4]

$$\begin{aligned}\sigma_\gamma &: \mathbb{R}^d \to \mathbb{R}^{d\times d}, \quad \sigma_\gamma(x) = [\gamma_{\sigma,1}x|\ldots|\gamma_{\sigma,d}x] + \gamma_{\sigma,d+1},\\ \mu_\gamma &: \mathbb{R}^d \to \mathbb{R}^d, \quad \mu_\gamma(x) = \gamma_{\mu,1}x + \gamma_{\mu,2},\end{aligned}$$

are affine-linear functions, which are determined by parameter tuples of matrices and vectors

$$\gamma_\sigma \in D_\sigma \subseteq (\mathbb{R}^{d\times d})^{d+1}, \quad \gamma_\mu \in D_\mu \subseteq \mathbb{R}^{d\times d}\times\mathbb{R}^d.$$

The parameter sets D_σ and D_μ are chosen to be compact. Together with a suitable compact parameter set $D_\varphi \subseteq \mathbb{R}^k$ for the initial function φ_γ, we obtain

$$\gamma := (\gamma_\sigma, \gamma_\mu, \gamma_\varphi) \in D_\sigma \times D_\mu \times D_\varphi := D \subseteq (\mathbb{R}^{d\times d})^{d+1} \times (\mathbb{R}^{d\times d}\times\mathbb{R}^d)\times\mathbb{R}^k.$$

This leads to an input dimension of our neural network Φ of

$$\dim_{\text{in}}(\Phi) = \dim(D\times[v,w]^d\times[0,T]) = d^3 + 2d^2 + 2d + 1 + k.$$

Kolmogorov PDEs with affine-linear coefficient functions regularly appear in applications; the heat equation from physics and the classical and generalized Black-Scholes equations from computational finance are important examples of Kolmogorov PDEs with affine-linear coefficient maps [13, 44]. Note that while affine-linear coefficient functions are important in practice, computationally fast to evaluate, and easy to parametrize, the presented method is not restricted to the case of affine-linear coefficients and can as well be used in a substantially more general setting.

3 Numerical Results

We implemented the framework described in Section 2 in PyTorch [42] and computed our results on a Nvidia DGX-1 using Tune [38] for experiment execution and hyperparameter optimization. In this section, we describe our setting and present four encouraging demonstrations of the performance of our algorithm.[5]

For the neural network Φ we propose a *Multilevel architecture* which is inspired by multilevel techniques such as Multilevel Monte Carlo methods [18], network architectures in [26, 59], and the architecture for the squaring function used in the proofs of our theoretical results in Section A.1, see also [58, Figure 2c]. One can view the output of the network Φ as a sum $\sum_{l=0}^{L-1}\Phi_l$ of sub-networks

[4] We denote by $[a_1|\ldots|a_d] \in \mathbb{R}^{d\times d}$ the horizontal concatenation of the vectors $a_1,\ldots,a_d \in \mathbb{R}^d$.

[5] For the implementation details we refer the reader to Section A.2 in the appendix and the repository associated with this work on `https://github.com/juliusberner/deep_kolmogorov`.

Φ_l with 2^l layer each. We think of the shallow network Φ_0 as computing a coarse approximation of $\bar{u}$ and of the deep networks Φ_l (with $l \geq 1$) as approximately learning the residuals $\bar{u} - \sum_{i=0}^{l-1} \Phi_i$. To facilitate optimization we normalize our inputs[6] and to enhance expressivity we add an initial layer which increases the width by a given factor q. The Multilevel architecture is depicted in Figure 2 and described in detail in Definition A.2 in the appendix.

Our optimized hyperparameters as well as an ablation study of our architecture and training scheme can be found in Sections A.2 and A.3 in the appendix. For all our experiments we were able to stick to a similar setup which depicts its robustness and general applicability. This is also mirrored by the small standard deviations of our considered errors across independent runs in Tables 1, 2, 3, and 4. These tables report average runtimes (in seconds), average approximation errors, and their standard deviations w.r.t. 4 independent runs each 4000 gradient descent steps. As an evaluation metric we approximately computed $\mathcal{L}^1$-errors via Monte Carlo sampling, that is

$$\left\| \frac{\Phi(\Lambda) - \bar{u}(\Lambda)}{1 + |\bar{u}(\Lambda)|} \right\|_{\mathcal{L}^1} := \mathbb{E}\left[\frac{|\Phi(\Lambda) - \bar{u}(\Lambda)|}{1 + |\bar{u}(\Lambda)|} \right] \approx \frac{1}{n} \sum_{i=1}^{n} \frac{|\Phi(\lambda_i) - \bar{u}(\lambda_i)|}{1 + |\bar{u}(\lambda_i)|} \tag{9}$$

with $n \in \mathbb{N}$ realizations $(\lambda_i)_{i=1}^n$ of i.i.d. samples drawn from the distribution of Λ (drawn independently of the training data in (7) and drawn independently for each evaluation step). This means that we always evaluate our model w.r.t. to the parametric solution map $\bar{u}$ on unseen input data; if no closed-form solution for $\bar{u}$ is available, as in the case of the Basket option in Section 3.2 below, we approximate $\bar{u}(\lambda_i)$ pointwise via Monte Carlo sampling, i.e.

$$\bar{u}(\lambda_i) = \bar{u}(\gamma_i, x_i, t_i) = \mathbb{E}[\varphi_{\gamma_i}(S_{\gamma_i, x_i, t_i})] \approx \frac{1}{m} \sum_{j=1}^{m} \varphi_{\gamma_i}(s_j) \tag{10}$$

where $(s_j)_{j=1}^m$ are realizations of i.i.d. samples drawn from the distribution of the Euler-Maruyama approximation $S_{\lambda_i}^{M,M}$ (drawn independently of the training data in (7) and drawn independently for each point and each evaluation step). Note that (9) is invariant under scaling of the hypercubes and locally corresponds to relative errors where the solution $\bar{u}$ is large and absolute errors where it is small, which in particular prevents division by zero.

3.1 Black-Scholes Options Pricing Model

Our first example shows that neural networks are capable of learning a parametric version of the highly-celebrated Black-Scholes option pricing model [9]. We consider a European put option which gives its owner the right, but not the obligation, to sell a single underlying financial asset at a specified strike price and at a given time. Formally, this corresponds to $d = 1$ and

$$\sigma_\gamma(x) = \gamma_\sigma x, \quad \mu_\gamma(x) = 0, \quad \varphi_\gamma(x) = \max\{\gamma_\varphi - x, 0\}, \quad x \in \mathbb{R},$$

with[7] $\gamma_\sigma \in D_\sigma \subseteq \mathbb{R}$ and $\gamma_\varphi \in D_\varphi \subseteq \mathbb{R}$. Effectively, this leads to an input dimension of our neural network Φ of $\dim_{\text{in}}(\Phi) = 4$. In case of the present Black-Scholes model, the associated SDE in (5) can actually be solved explicitly; it gives rise to geometric Brownian motion with uniformly distributed volatility $\Gamma_\sigma \in D_\sigma$, initial value $X \in [v, w]$, and stopping time $\mathcal{T} \in [0, T]$, i.e.

$$S_\Lambda = X e^{-0.5\mathcal{T}\Gamma_\sigma^2 + \sqrt{\mathcal{T}}\Gamma_\sigma N}$$

where $N \sim \mathcal{N}(0,1)$ is normally distributed and independent of Λ. We thus obtain an analytic expression for the parametric PDE solution,

$$\bar{u}(\gamma, x, t) = \gamma_\varphi \Psi(h_\gamma(x,t) + \sqrt{t}\,\gamma_\sigma) - x\Psi(h_\gamma(x,t)),$$

and the partial derivatives, e.g.

$$\frac{\partial \bar{u}}{\partial \gamma_\sigma}(\gamma, x, t) = x\sqrt{t}\Psi'(-h_\gamma(x,t)),$$

where

$$\Psi(z) := \tfrac{1}{2}\big(1 + \operatorname{erf}\big(\tfrac{z}{\sqrt{2}}\big)\big) \quad \text{and} \quad h_\gamma(x,t) := -\tfrac{1}{\sqrt{t}\,\gamma_\sigma}\big(\ln\big(\tfrac{x}{\gamma_\varphi}\big) + \tfrac{t\gamma_\sigma^2}{2}\big),$$

see [4, Section 13.7]. This analytic expression can be used to evaluate the performance of our algorithm. We point out that the partial derivatives of $\bar{u}$ are crucial in option pricing and each of

[6] We know the underlying (uniform) distribution and therefore can normalize each input in an exact manner.

[7] Note that $\sigma_\gamma = \sigma_{\gamma,1}$ in the formal framework described in Section 2.1 but here and in the following we use the natural identifications, e.g. $D_\sigma \cong D_\sigma \times \{0\}$.

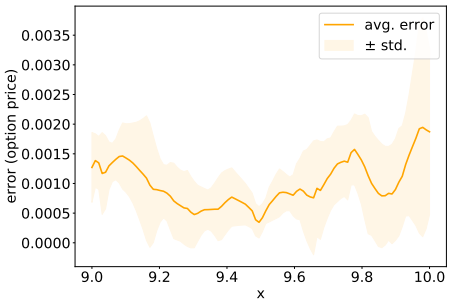


Figure 3: Shows the average prediction error $\frac{|\Phi(\gamma,\cdot,t)-\bar{u}(\gamma,\cdot,t)|}{1+|\bar{u}(\gamma,\cdot,t)|}$ and its standard deviation at $t = 0.5$, $\gamma_\sigma = 0.35$, and $\gamma_\varphi = 11$.

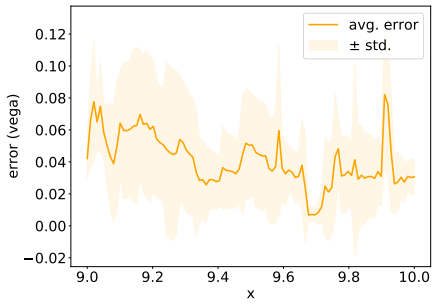


Figure 4: Shows the average error of the Vega $\frac{|\frac{\partial\Phi}{\partial\gamma_\sigma}(\gamma,\cdot,t)-\frac{\partial\bar{u}}{\partial\gamma_\sigma}(\gamma,\cdot,t)|}{1+|\frac{\partial\bar{u}}{\partial\gamma_\sigma}(\gamma,\cdot,t)|}$ and its standard deviation at $t = 0.5$, $\gamma_\sigma = 0.35$, and $\gamma_\varphi = 11$.

them is associated with a distinct economic interpretation. They are often referred to as *Greeks* and describe the sensitivity of the option price w.r.t. different model parameters, see for instance [4, 50]. The most prominent Greeks are given by

$$\Delta = \frac{\partial\bar{u}}{\partial x}, \quad \text{Vega} = \frac{\partial\bar{u}}{\partial\gamma_\sigma}, \quad \Theta = -\frac{\partial\bar{u}}{\partial t}.$$

On the basis of the proposed algorithm, our neural network Φ is capable of learning the parametric solution map $\bar{u}$ of the above problem in 24000 gradient updates up to an average $\mathcal{L}^1$-error of 0.0011, see Table 1 and Figure 3. As expected, the partial derivatives of our network Φ (computed via automatic differentiation) approximate the partial derivatives of $\bar{u}$ as can be seen in Figure 4. Further evidence can be found in Figures 5, 6, 7, and 8 in the appendix. Even though the parametric PDE problem can be solved explicitly in this special case, we use this relatively simple example for the purpose of illustrating our algorithm in an intuitive setting. As we will see below, the proposed algorithm is by no means restricted to such basic examples and can be applied successfully to much more complex and high-dimensional problems as well.

3.2 Basket Put Option

In the following we show that we can obtain comparable results to Section 3.1 in the case of a considerably more complicated Basket put option pricing problem, where analytical solutions of the PDE and the SDE are lacking. In such cases, our algorithm allows practitioners to nevertheless gain valuable insights into the behaviour of the PDE solution manifold as input parameters vary. By means of our trained model Φ one can easily compute sensitivity values $\frac{\partial\Phi}{\partial\gamma} \approx \frac{\partial\bar{u}}{\partial\gamma}$, $\frac{\partial\Phi}{\partial t} \approx \frac{\partial\bar{u}}{\partial t}$, and $\frac{\partial\Phi}{\partial x} \approx \frac{\partial\bar{u}}{\partial x}$ via automatic differentiation or fit the parameter γ to a real-world data-set $((x_i,t_i),u_\gamma(x_i,t_i))_{i=1}^m$ with unknown γ by minimizing $\min_{\gamma\in D}\sum_{i=1}^m\big(\Phi(\gamma,x_i,t_i)-u_\gamma(x_i,t_i)\big)^2$ via stochastic gradient descent. Moreover, one can obtain estimates for probabilistic quantities related to uncertainty such as

$$\mathbb{V}[u_\Xi(x,t)] \approx \mathbb{V}[\Phi(\Xi,x,t)] \approx \tfrac{1}{m-1}\sum_{i=1}^m\big(\Phi(\xi_i,x,t)-\tfrac{1}{n}\sum_{j=1}^m\Phi(\xi_j,x,t)\big)^2$$

where $(\xi_i)_{i=1}^m$ are realizations of i.i.d. samples drawn from the distribution of a random variable Ξ of interest. None of these types of insights were accessible before the presented deep learning method.

We proceed by demonstrating the performance of the proposed algorithm for a general multidimensional affine-linear setting as described in Section 2.1. To this end, let $d = 3$ and define the initial condition via

$$\varphi_\gamma(x) = \max\big\{\gamma_\varphi - \tfrac{1}{3}\textstyle\sum_{i=1}^3 x_i, 0\big\}, \quad x\in\mathbb{R}^3,$$

with $\gamma_\varphi \in D_\varphi \subseteq \mathbb{R}$. This corresponds to the situation of a Basket put option in a very general multidimensional Black-Scholes model with 3 potentially highly correlated assets. Note that within the above setup, the input dimension of our neural network Φ is given by

$$\dim_{\text{in}}(\Phi) = d^3 + 2d^2 + 2d + 1 + 1 = 53.$$

To generate samples of our target variable $\varphi_\Gamma(S_\Lambda)$, we simulate solutions of the SDE in (5) using the Euler-Maruyama scheme (8) with $M = 25$ equidistant steps. Moreover, we use a Monte Carlo

Table 1: Results for the Black-Scholes model

step	avg. time (s)	avg. $\mathcal{L}^1$-error
0	0 ± 0	0.6812 ± 0.0704
4k	471 ± 3	0.0088 ± 0.0056
8k	943 ± 6	0.0062 ± 0.0025
12k	1413 ± 9	0.0030 ± 0.0004
16k	1885 ± 11	0.0017 ± 0.0001
20k	2356 ± 14	0.0013 ± 0.0002
24k	2827 ± 17	0.0011 ± 0.0001

Table 2: Results for the Basket option

step	avg. time (s)	avg. $\mathcal{L}^1$-error
0	0 ± 0	0.7912 ± 0.0276
4k	811 ± 7	0.0131 ± 0.0019
8k	1614 ± 4	0.0087 ± 0.0013
12k	2434 ± 28	0.0062 ± 0.0009
16k	3236 ± 27	0.0058 ± 0.0011
20k	4162 ± 154	0.0046 ± 0.0007
24k	5077 ± 291	0.0042 ± 0.0002
28k	6024 ± 463	0.0039 ± 0.0001

approximation with $m = 2^{20}$ samples to compute the pointwise evaluation of the reference solution $\bar{u}(\lambda_i)$ according to (10) as needed for the error estimation in (9). Despite the considerably higher dimension of this problem compared with the previous problem from Section 3.1, our deep learning approach shows almost the same approximation behavior, see Table 2. This remarkably weak dependence on the dimension of the input data is further supported by the next examples from physical modelling, where we shall increase the dimensionality of the studied problems even further.

3.3 Heat Equation with Varying Diffusion Coefficients

In this Section, we present two examples of high-dimensional heat equations in $d = 10$ and $d = 150$ dimensions with paraboloid and Gaussian initial conditions

$$\varphi_\gamma(x) = \|x\|^2 \quad \text{(paraboloid)} \quad \text{and} \quad \varphi_\gamma = e^{-\|x\|^2} \quad \text{(Gaussian)}.$$

This formally corresponds to

$$\sigma_\gamma(x) = \gamma_\sigma \quad \text{and} \quad \mu_\gamma(x) = 0$$

where we use a matrix $\gamma_\sigma \in D_\sigma \subseteq \mathbb{R}^{10\times 10}$ for the paraboloid case and a scalar $\gamma_\sigma \in D_\sigma \subseteq \mathbb{R}$ in the Gaussian case, leading to input dimensions of our models Φ of

$$\dim_{\text{in}}(\Phi) = d^2 + d + 1 = 111 \quad \text{(paraboloid)} \quad \text{and} \quad \dim_{\text{in}}(\Phi) = d + 1 + 1 = 152 \quad \text{(Gaussian)}.$$

Notice that here the solution of the corresponding SDE can be directly sampled via a Brownian motion with uniformly distributed scaling Γ_σ, initial position X, and stopping time $\mathcal{T}$, i.e.

$$S_\Lambda = X + \sqrt{\mathcal{T}}\,\Gamma_\sigma N$$

where $N \sim \mathcal{N}(0, I_d)$ is normally distributed and independent of Λ, see [5, Section 3.2]. For evaluation purposes, these examples were purposefully constructed to have analytic expressions for the parametric solution maps $\bar{u}$, which are given by

$$\bar{u}(\gamma_\sigma, x, t) = \|x\|^2 + t\,\mathrm{Trace}(\gamma_\sigma\gamma_\sigma^*) \quad \text{(paraboloid)}, \quad \bar{u}(\gamma_\sigma, x, t) = \frac{e^{-\frac{\|x\|^2}{1+2t\gamma_\sigma^2}}}{(1+2t\gamma_\sigma^2)^{d/2}} \quad \text{(Gaussian)}.$$

However, in almost all other practical cases an analytic solution for $\bar{u}$ is impossible to obtain and numerical methods are the only path forward.

The above dimensionality settings represent regimes which are completely out of scope for all preexisting numerical schemes. Nevertheless, Tables 3 and 4 confirm that our proposed deep learning method once again efficiently converges to the desired parametric solution map $\bar{u}$. Our results empirically demonstrate that, contrary to conventional numerical solvers, our deep learning based method does not suffer from the curse of dimensionality, see also Figure 9 in the appendix. We will rigorously prove this fact in the next section.

4 Theoretical Guarantees

As a first example, we stick to the heat equation with paraboloid initial condition from above and show that neural networks are capable of simultaneously approximating the parametric solution map $\bar{u}$ and its gradient with the number of network parameters scaling only polynomially in the dimension d, see Theorem A.3 in the appendix for a proof. Such an approximation guarantee without curse of dimensionality ensures that sensitivity analysis is possible even in very high dimensions.

Table 3: Results for the heat equation with paraboloid initial condition

step	avg. time (s)	avg. $\mathcal{L}^1$-error
0	0 ± 0	0.9609 ± 0.0052
4k	1904 ± 19	0.0150 ± 0.0008
8k	3808 ± 37	0.0120 ± 0.0007
12k	5712 ± 57	0.0093 ± 0.0006
16k	7616 ± 76	0.0068 ± 0.0001
20k	9520 ± 95	0.0062 ± 0.0003
24k	11424 ± 114	0.0057 ± 0.0001
28k	13328 ± 132	0.0056 ± 0.0000

Table 4: Results for the heat equation with Gaussian initial condition

step	avg. time (s)	avg. $\mathcal{L}^1$-error
0	0 ± 0	0.2035 ± 0.0714
4k	2070 ± 40	0.0123 ± 0.0047
8k	4131 ± 82	0.0050 ± 0.0018
12k	6192 ± 124	0.0051 ± 0.0022
16k	8258 ± 165	0.0033 ± 0.0015
20k	10323 ± 206	0.0025 ± 0.0011
24k	12388 ± 247	0.0024 ± 0.0008
28k	14454 ± 290	0.0019 ± 0.0002

Theorem 3 (Sobolev Approximation)**.** *There exists a neural network Φ with ReLU activation function and $\mathcal{O}(d^4 \log(d/\varepsilon))$ parameters satisfying that*

$$\|\Phi - \bar{u}\|_{\mathcal{L}^\infty(D\times[v,w]^d\times[0,T])} \leq \varepsilon \quad \textit{and} \quad \|\nabla\Phi - \nabla\bar{u}\|_{\mathcal{L}^\infty(D\times[v,w]^d\times[0,T])} \leq \varepsilon.$$

Let us now consider the heat equation with varying diffusivity and Gaussian initial condition. In fact, our framework allows us to rigorously prove sample complexity estimates for this problem which represents an almost unique scenario for deep learning based methods. This is rendered possible by the structure of the underlying parametric Kolmogorov PDE and its associated SDE which allows us to describe the distribution of the predictor and target variable, simulate i.i.d. samples, and infer regularity properties on the regression function. We briefly sketch the theorem in the following; the precise formulation and the proof is given in Theorem A.5 in the appendix.

Theorem 4 (Generalization)**.** *Using $s \sim (d/\varepsilon)^2 \operatorname{polylog}(d/\varepsilon)$ many samples, every empirical risk minimizer $\hat{\Phi}$ of (7) in a suitable hypothesis space $\mathcal{H}$ of neural networks with ReLU activation function, $\mathcal{O}(\operatorname{polylog}(d/\varepsilon))$ layers, $\mathcal{O}(d)$ neurons per layer, and parameters bounded by $\mathcal{O}(1)$ satisfies with high probability that*

$$\tfrac{1}{V}\|\hat{\Phi} - \bar{u}\|^2_{\mathcal{L}^2(D\times[v,w]^d\times[0,T])} \leq \varepsilon$$

where $V := \operatorname{vol}(D \times [v,w]^d \times [0,T])$.

Note that it holds that

$$\tfrac{1}{V}\|\cdot\|^2_{\mathcal{L}^2(D\times[v,w]^d\times[0,T])} = \|\cdot\|^2_{\mathcal{L}^2(\mathbb{P}_\Lambda)}$$

where $\mathbb{P}_\Lambda$ is the uniform probability measure on $D \times [v,w]^d \times [0,T]$. Thus the estimate in Theorem 4 can be viewed as an estimate in the space $\mathcal{L}^2(\mathbb{P}_\Lambda)$ and we want to emphasize that our setting easily allows us to choose arbitrary probability measures $\mathbb{P}$ on $D \times [v,w]^d \times [0,T]$ and prove analogous results w.r.t. the $\mathcal{L}^2(\mathbb{P})$-norm.

5 Conclusion

The method introduced in this paper is the first deep learning algorithm for the numerical solution of parametric Kolmogorov PDEs and one of few existing algorithms whose use is computationally tractable in high-dimensional settings. The parametric nature of our approach readily allows for sensitivity analysis, model calibration, and uncertainty quantification, all which is of high interest in a variety of applications. Successful numerical experiments in both low- and high-dimensional settings empirically confirm the functionality of the proposed algorithm. In addition, we are able to provide theoretical guarantees for the applicability of our method in high-dimensions.

Besides solving an important problem in scientific computing, our work introduces a class of learning problems that allows for the rigorous investigation of expressivity and sample complexity, along with stable and interpretable algorithms. Such strong results become possible by leveraging the mathematical structure of the learning problem associated with the parametric PDE. We anticipate that the formulation and study of such structured problems will constitute an important future direction of research in the scientific machine learning community as it can enable reliable and interpretable algorithms for the solution of previously intractable problems: in our case parametric families of Kolmogorov PDEs. This contributes substantially to areas like physical modelling of diffusion processes and computational finance, which all rely on the use of such PDEs.

Broader Impact

The deep-learning technique presented in this work is the first computationally scalable method for the numerical solution of high-dimensional parametric Kolmogorov PDEs. It is also the first method which allows for a straightforward sensitivity analysis of the associated high-dimensional PDE solution manifold with respect to input parameters. In addition, it newly allows for high-dimensional data-driven model calibration and uncertainty quantification. While it is a difficult task to precisely estimate the cascading effects of technological innovations on wider society, it is reasonable to assume that the ubiquity of Kolmogorov equations in science and engineering will lead to a positive impact of our new findings on a multitude of technical areas of social importance.

As an example, Kolmogorov PDEs are heavily used in physics for the modelling of heat flow and diffusion processes [41, 56]. Simultaneously, Fokker-Planck equations, which take the form of Kolmogorov equations in particular special cases, are used in the geophysical and atmospheric sciences as modelling tools for climate change projections [25, 52]. Our described algorithm has clear promise to make previously intractable high-dimensional physical models computationally accessible to scientists. Additionally, our method allows for an easy investigation of changes in complex model forecasts as input parameters are varied during sensitivity analysis. Such advancements have the potential to accelerate scientific research and can directly lead to better predictive models in applied physics and engineering. Reliable and efficient predictive models in turn are essential to rationally inform public policy.

A conceivable risk posed by our work might come in the form of the uncritical use of our algorithm in applications related to financial engineering. The Black-Scholes equation and associated models have been notoriously misused in the last decades by semi-technical users working in financial sectors around the world [31, 57]. The naive usage of technical tools in computational finance has thus likely been a contributing factor to periods of economic instability in recent history. Our technique can now add a powerful solver for high-dimensional parametric PDE problems to the tool kits of individual end-users in finance with various degrees of scientific expertise. Inexperienced users without appropriate quantitative background might be prone to erroneously taking the complexity of a high-dimensional financial model as an indicator for its accuracy. Therefore, one must take great care to systematically inform users without suitable experience in such a scenario that merely increasing the dimension of an inadequate financial model might not necessarily make its results more accurate.

In total, we are confident that the net impact of our work on the scientific community as well as broader society is positive. The probability of uncritical use of our technique and other algorithms in financial engineering can likely be substantially mitigated by targeted educational interventions and we would encourage practical research in this direction. At the same time, we note that our technical contribution is a general-purpose tool which has the potential to stimulate the acceleration of scientific progress in a wide variety of disciplines.

Acknowledgments and Disclosure of Funding

The research of Julius Berner was supported by the Austrian Science Fund (FWF) under grant I3403-N32. The research of Markus Dablander was supported by the UK EPSRC Centre For Doctoral Training in Industrially Focused Mathematical Modelling (EP/L015803/1).

References

[1] C. D. Aliprantis and K. C. Border. Infinite dimensional analysis: a hitchhiker's guide, 2006.

[2] W. F. Ames. *Numerical methods for partial differential equations*. Academic press, 2014.

[3] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv:1607.06450*, 2016.

[4] P. Baldi. *Stochastic Calculus: An Introduction Through Theory and Exercises*. Universitext. Springer International Publishing, 2017.

[5] C. Beck, S. Becker, P. Grohs, N. Jaafari, and A. Jentzen. Solving stochastic differential equations and Kolmogorov equations by means of deep learning. *arXiv:1806.00421*, 2018.

[6] C. Beck, E. Weinan, and A. Jentzen. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *Journal of Nonlinear Science*, 29(4):1563–1619, 2019.

[7] J. Berner, D. Elbrächter, P. Grohs, and A. Jentzen. Towards a regularity theory for ReLU networks – chain rule and global error estimates. In *2019 13th International conference on Sampling Theory and Applications (SampTA)*, pages 1–5, 2019.

[8] J. Berner, P. Grohs, and A. Jentzen. Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black–Scholes partial differential equations. *SIAM Journal on Mathematics of Data Science*, 2(3):631–657, 2020.

[9] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654, 1973.

[10] M. G. Crandall, H. Ishii, and P.-L. Lions. User's guide to viscosity solutions of second order partial differential equations. *Bulletin of the American mathematical society*, 27(1):1–67, 1992.

[11] F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39(1):1–49, 2002.

[12] M. Eigel, R. Schneider, P. Trunschke, and S. Wolf. Variational Monte Carlo-bridging concepts of machine learning and high dimensional partial differential equations. *arXiv:1810.01348*, 2018.

[13] E. Ekström and J. Tysk. The Black–Scholes equation in stochastic volatility models. *Journal of Mathematical Analysis and Applications*, 368(2):498 – 507, 2010.

[14] D. Elbrächter, P. Grohs, A. Jentzen, and C. Schwab. DNN expression rate analysis of high-dimensional PDEs: Application to option pricing. *arXiv:1809.07669*, 2018.

[15] L. C. Evans and R. F. Gariepy. *Measure Theory and Fine Properties of Functions, Revised Edition*. Textbooks in Mathematics. CRC Press, 2015.

[16] A. Friedman. *Stochastic Differential Equations and Applications*. Dover Books on Mathematics. Dover Publications, 2012.

[17] J. Gall. *Brownian Motion, Martingales, and Stochastic Calculus*. Graduate Texts in Mathematics. Springer International Publishing, 2016.

[18] M. B. Giles. Multilevel Monte Carlo methods. *Acta Numerica*, 24:259–328, 2015.

[19] C. Graham and D. Talay. *Stochastic Simulation and Monte Carlo Methods: Mathematical Foundations of Stochastic Simulation*. Stochastic Modelling and Applied Probability. Springer Berlin Heidelberg, 2013.

[20] P. Grohs, F. Hornung, A. Jentzen, and P. Von Wurstemberger. A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations. *arXiv:1809.02362*, 2018.

[21] P. Grohs, D. Perekrestenko, D. Elbrächter, and H. Bölcskei. Deep Neural Network Approximation Theory. *arxiv:1901.02220*, 2019.

[22] I. Gühring, G. Kutyniok, and P. Petersen. Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms. *Analysis and Applications*, pages 1–57, 2019.

[23] M. Hairer, M. Hutzenthaler, A. Jentzen, et al. Loss of regularity for Kolmogorov equations. *The Annals of Probability*, 43(2):468–527, 2015.

[24] J. Han, A. Jentzen, and E. Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[25] K. Hasselmann. Stochastic climate models part i. theory. *tellus*, 28(6):473–485, 1976.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] J. S. Hesthaven and S. Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, 2018.

[28] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[29] M. Hutzenthaler, A. Jentzen, T. Kruse, and T. A. Nguyen. A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations. *arXiv:1901.10854*, 2019.

[30] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015.

[31] R. A. Jarrow. Risk management models: construction, testing, usage. *The Journal of Derivatives*, 18(4):89–98, 2011.

[32] A. Jentzen, D. Salimova, and T. Welti. A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients. *arXiv:1809.07321*, 2018.

[33] Y. Khoo, J. Lu, and L. Ying. Solving parametric PDE problems with artificial neural networks. *arXiv:1707.03351*, 2017.

[34] P. E. Kloeden and E. Platen. *Numerical solution of stochastic differential equations*, volume 23 of *Applications of Mathematics (New York)*. Springer-Verlag, Berlin, 1992.

[35] G. Kutyniok, P. Petersen, M. Raslan, and R. Schneider. A theoretical analysis of deep neural networks and parametric PDEs. *arXiv:1904.00377*, 2019.

[36] F. Laakmann and P. Petersen. Efficient approximation of solutions of parametric linear transport equations by ReLU DNNs. *arXiv:2001.11441*, 2020.

[37] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, and A. Talwalkar. Massively parallel hyperparameter tuning. *arXiv:1810.05934*, 2018.

[38] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica. Tune: A research platform for distributed model selection and training. *arXiv:1807.05118*, 2018.

[39] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv:1711.05101*, 2017.

[40] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

[41] A. Pascucci. Kolmogorov equations in physics and in finance. In *Elliptic and parabolic problems*, pages 353–364. Springer, 2005.

[42] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[43] P. Petersen and F. Voigtlaender. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Networks*, 108:296–330, 2018.

[44] O. Pironneau and Y. Achdou. Partial differential equations for option pricing. *Handbook of Numerical Analysis*, 15:369–495, 2009.

[45] D. Pollard. *A User's Guide to Measure Theoretic Probability*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2002.

[46] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[47] C. Reisinger and Y. Zhang. Rectified deep neural networks overcome the curse of dimensionality for nonsmooth value functions in zero-sum games of nonlinear stiff systems. *arXiv:1903.06652*, 2019.

[48] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747*, 2016.

[49] C. Schwab and J. Zech. Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ. *Analysis and Applications*, 17(01):19–55, 2019.

[50] R. Seydel. *Tools for computational finance*, volume 3. Springer, 2006.

[51] J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[52] J. Thuburn. Climate sensitivities via a fokker–planck adjoint approach. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 131(605):73–92, 2005.

[53] V. Vapnik. *Statistical learning theory. 1998*, volume 3. Wiley, New York, 1998.

[54] E. Weinan and B. Yu. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[55] E. Weinan, J. Han, and A. Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

[56] D. V. Widder. *The heat equation*, volume 67. Academic Press, 1976.

[57] P. Wilmott. The use, misuse and abuse of mathematics in finance. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 358(1765):63–73, 2000.

[58] D. Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94: 103–114, 2017.

[59] F. Yu, D. Wang, E. Shelhamer, and T. Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412, 2018.

A Appendix

A.1 Theoretical Results

First we state our assumptions on the coefficient maps and initial conditions.

Assumptions A.1 (Coefficient Maps & Initial Conditions)**.** *Let D be a compact set in Euclidean space and for every $\gamma \in D$ let $\varphi_\gamma \in \mathcal{C}(\mathbb{R}^d, \mathbb{R})$, $\sigma_\gamma \in \mathcal{C}(\mathbb{R}^d, \mathbb{R}^{d\times d})$, and $\mu_\gamma \in \mathcal{C}(\mathbb{R}^d, \mathbb{R}^d)$. Assume that for every $x \in \mathbb{R}^d$ the mappings*

$$\gamma \mapsto \varphi_\gamma(x), \quad \gamma \mapsto \sigma_\gamma(x), \quad \text{and} \quad \gamma \mapsto \mu_\gamma(x)$$

are continuous and that there exists $c \in (0,\infty)$ such that for every $\gamma \in D$, $x, y \in \mathbb{R}^d$ it holds that[8]

(i) $|\varphi_\gamma(x) - \varphi_\gamma(y)| \leq c\|x-y\|(1 + \|x\|^c + \|y\|^c)$,

(ii) $\|\mu_\gamma(x) - \mu_\gamma(y)\| + \|\sigma_\gamma(x) - \sigma_\gamma(y)\| \leq c\|x-y\|$, *and*

(iii) $|\varphi_\gamma(0)| + \|\mu_\gamma(0)\| + \|\sigma_\gamma(0)\| \leq c$.

Note that the continuity assumptions on σ_γ and μ_γ and the condition in Item (ii) are fulfilled for the case of affine-linear coefficient functions as described in Section 2.1 and used in our examples. Further, the polynomial growth condition on the local Lipschitz constant in Item (i), the uniform bound in Item (iii), and the continuity assumption on φ_γ are also satisfied for all our considered examples. Under these assumptions we can precisely formulate the setting we are working in.

Definition A.1 (Parametric Kolmogorov PDEs)**.** *For every $\gamma \in D$ let $u_\gamma \colon \mathbb{R}^d \times [0,\infty) \to \mathbb{R}$ be the unique continuous, at most polynomially growing function satisfying for every $x \in \mathbb{R}^d$ that $u_\gamma(x,0) = \varphi_\gamma(x)$ and satisfying that $u|_{\mathbb{R}^d\times(0,\infty)}$ is a viscosity solution of the Kolmogorov PDE*

$$\tfrac{\partial u_\gamma}{\partial t}(x,t) = \tfrac{1}{2}\operatorname{Trace}\big(\sigma_\gamma(x)[\sigma_\gamma(x)]^*(\nabla_x^2 u_\gamma)(x,t)\big) + \langle \mu_\gamma(x), (\nabla_x u_\gamma)(x,t)\rangle$$

for $(x,t) \in \mathbb{R}^d \times (0,\infty)$, see [23, Corollary 4.17]. Let $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in[0,T]}, \mathbb{P})$ be a suitable filtered probability space satisfying the usual conditions, let

$$(B_t)_{t\geq 0} \colon [0,\infty) \times \Omega \to \mathbb{R}^d \tag{11}$$

be a standard d-dimensional $(\mathcal{F}_t)$-Brownian motion, let $T \in (0,\infty)$, $v \in \mathbb{R}$, $w \in (v,\infty)$ and let

$$\Lambda = (\Gamma, X, \mathcal{T}) \colon \Omega \to D \times [v,w]^d \times [0,T]$$

be a $\mathcal{F}_0$-measurable, uniformly distributed random variable. Let

$$(S_{\gamma,x,t})_{t\geq 0} \colon [0,\infty)\times\Omega \to \mathbb{R}^d, \quad (\gamma,x) \in D \times \mathbb{R}^d, \quad \text{and} \quad (S_{\Gamma,X,t})_{t\geq 0} \colon [0,\infty)\times\Omega \to \mathbb{R}^d$$

be the up to indistinguishability unique $(\mathcal{F}_t)$-adapted stochastic processes with continuous sample paths satisfying that for every $(\gamma,x,t) \in D \times \mathbb{R}^d \times [0,\infty)$ it holds $\mathbb{P}$-a.s. that

$$S_{\gamma,x,t} = x + \int_0^t \mu_\gamma(S_{\gamma,x,s})ds + \int_0^t \sigma_\gamma(S_{\gamma,x,s})dB_s, \tag{12}$$

and that for every $t \in [0,\infty)$ it holds $\mathbb{P}$-a.s. that

$$S_{\Gamma,X,t} = X + \int_0^t \mu_\Gamma(S_{\Gamma,X,s})ds + \int_0^t \sigma_\Gamma(S_{\Gamma,X,s})dB_s, \tag{13}$$

see, for instance, [17, Proof of Theorem 8.3]. For every $M \in \mathbb{N}$, $(\gamma,x,t) \in D \times \mathbb{R}^d \times [0,\infty)$ let

$$(S^{M,m}_{\gamma,x,t})_{m=0}^M \colon \{0,\dots,M\} \times \Omega \mapsto \mathbb{R}^d$$

be a stochastic process satisfying that $S^{M,0}_{\gamma,x,t} = x$ and for every $m \in \{0,\dots,M-1\}$ that

$$S^{M,m+1}_{\gamma,x,t} = S^{M,m}_{\gamma,x,t} + \mu_\gamma(S^{M,m}_{\gamma,x,t})\tfrac{t}{M} + \sigma_\gamma(S^{M,m}_{\gamma,x,t})\big(B_{\frac{(m+1)t}{M}} - B_{\frac{mt}{M}}\big)$$

[8] For a finite index set I and $a, b \in \mathbb{R}^I$ we define $\|a\| = \sqrt{\sum_{i\in I}|a_i|^2}$ and $\langle a,b\rangle = \sum_{i\in I} a_i b_i$.

and for every $M \in \mathbb{N}$ let

$$(S^{M,m}_{\Gamma,X,\mathcal{T}})^M_{m=0} \colon \{0,\dots,M\} \times \Omega \mapsto \mathbb{R}^d$$

be a stochastic process satisfying that $S^{M,0}_{\Gamma,X,\mathcal{T}} = X$ and for every $m \in \{0,\dots,M-1\}$ that

$$S^{M,m+1}_{\Gamma,X,\mathcal{T}} = S^{M,m}_{\Gamma,X,\mathcal{T}} + \mu_\Gamma(S^{M,m}_{\Gamma,X,\mathcal{T}})\tfrac{\mathcal{T}}{M} + \sigma_\Gamma(S^{M,m}_{\Gamma,X,\mathcal{T}})\big(B_{\frac{(m+1)\mathcal{T}}{M}} - B_{\frac{m\mathcal{T}}{M}}\big).$$

Finally, let the random variable $Y \colon \Omega \mapsto \mathbb{R}$ be given by

$$Y := \varphi_\Gamma(S_\Lambda) = \varphi_\Gamma(S_{\Gamma,X,\mathcal{T}})$$

and for every $M \in \mathbb{N}$ let the random variable $Y^M \colon \Omega \mapsto \mathbb{R}$ be given by

$$Y^M := \varphi_\Gamma(S^{M,M}_\Lambda) = \varphi_\Gamma(S^{M,M}_{\Gamma,X,\mathcal{T}}).$$

In order to prove Theorem 1 we assume the following regularity on our SDEs in (12) and (13).

Assumptions A.2 (Regularity Assumptions)**.** *Assume that there exists a jointly measurable[9] function*

$$\Upsilon \colon \mathcal{C}([0,T],\mathbb{R}^d) \times D \times [v,w]^d \times [0,T] \to \mathbb{R}$$

such that it holds $\mathbb{P}$-a.s. that

$$\Upsilon(B,\Gamma,X,\mathcal{T}) = \varphi_\Gamma(S_\Lambda)$$

and for every $(\gamma,x,t) \in D \times [v,w]^d \times [0,T]$ it holds $\mathbb{P}$-a.s. that

$$\Upsilon(B,\gamma,x,t) = \varphi_\gamma(S_{\gamma,x,t}),$$

where $B \colon \Omega \to \mathcal{C}([0,T],\mathbb{R}^d)$, $\omega \mapsto (t \mapsto B_t(\omega))$, denotes the mapping to the sample paths of the Brownian motion in (11)*.*

Note that the above assumptions are satisfied for the Black-Scholes model in Section 3.1 and the heat equations in Section 3.3. In the former case we can write

$$\Upsilon(b,\gamma,x,t) = \max\{\gamma_\varphi - xe^{-0.5t\,\gamma_\sigma^2 + \sqrt{t}\,\gamma_\sigma b(1)}, 0\}$$

and in the latter

$$\Upsilon(b,\gamma,x,t) = \|x + \sqrt{t}\,\gamma_\sigma b(1)\|^2 \quad \text{(paraboloid)}, \qquad \Upsilon(b,\gamma,x,t) = e^{-\|x+\sqrt{t}\,\gamma_\sigma b(1)\|^2} \quad \text{(Gaussian)}$$

where $(b,\gamma,x,t) \in \mathcal{C}([0,T],\mathbb{R}^d) \times D \times [v,w]^d \times [0,T]$. Moreover, the existence of a suitable Υ is in general given for non-parametric Kolmogorov PDEs, see [17, Theorem 8.5] and [5]. First we establish that under our assumptions the minimizer of the statistical learning problem is indeed the parametric Kolmogorov PDE solution map.

Theorem A.1 (Learning Problem)**.** *Let Assumptions A.1 and A.2 be satisfied. Then it holds that*

$$\bar{u} : D \times [v,w]^d \times [0,T] \to \mathbb{R}, \quad (\gamma,x,t) \mapsto \bar{u}(\gamma,x,t) := u_\gamma(x,t)$$

is the (up to sets of Lebesgue measure zero) unique minimizer of the statistical learning problem

$$\min_f \mathbb{E}\Big[\big(f(\Lambda) - Y\big)^2\Big] \tag{14}$$

where the minimum is taken over all measurable functions $f \colon D \times [v,w]^d \times [0,T] \to \mathbb{R}$.

Proof. Note that one can extend standard results on the moments of SDE solution processes (see [34, Theorems 4.5.3 and 4.5.4] and [16, Chapter 5, Theorem 2.3]) to prove that S_Λ and thus also the target variable $Y = \varphi_\Gamma(S_\Lambda)$ have bounded moments. It is well-known that under this condition the (up to sets of measure zero w.r.t. the distribution of Λ) unique solution of the statistical learning problem (14) is given by the regression function

$$f^*(\gamma,x,t) := \mathbb{E}[Y \mid \Lambda = (\gamma,x,t)], \quad (\gamma,x,t) \in D \times [v,w]^d \times [0,T], \tag{15}$$

that is

$$f^* = \operatorname{argmin}_f \mathbb{E}\Big[\big(f(\Lambda) - Y\big)^2\Big],$$

[9] If not further specified, we consider measurability w.r.t. the corresponding Borel sigma algebras.

see, for instance, [11]. Moreover, the Feynman-Kac formula establishes for every $(\gamma, x, t) \in D \times [v, w]^d \times [0, T]$ that

$$\mathbb{E}[\varphi_\gamma(S_{\gamma,x,t})] = u_\gamma(x,t) = \bar{u}(\gamma, x, t), \tag{16}$$

see [23, Corollary 4.17]. Finally, Assumptions A.2 and the independence of B and Λ ensure that for every Borel measurable set $A \subseteq D \times [v, w]^d \times [0, T]$ it holds that

$$\begin{aligned}\mathbb{E}\big[\mathbf{1}_{\{\Lambda \in A\}} \varphi_\Gamma(S_\Lambda)\big] &= \int_A \int_{\mathcal{C}([0,T],\mathbb{R}^d)} \Upsilon(b, \gamma, x, t)\, d\mathbb{P}_B(b)\, d\mathbb{P}_{(\Gamma,X,\mathcal{T})}(\gamma, x, t) \\ &= \int_A \mathbb{E}\big[\varphi_\gamma(S_{\gamma,x,t})\big]\, d\mathbb{P}_{(\Gamma,X,\mathcal{T})}(\gamma, x, t)\end{aligned}$$

where we denote the distributions of Λ and B by $\mathbb{P}_{(\Gamma,X,\mathcal{T})}$ and $\mathbb{P}_B$ (Wiener measure), respectively. Together with the fact that Λ is uniformly distributed, this proves that for almost every $(\gamma, x, t) \in D \times [v, w]^d \times [0, T]$ it holds that

$$\mathbb{E}[\,Y |\, \Lambda = (\gamma, x, t)] = \mathbb{E}[\varphi_\Gamma(S_\Lambda) \mid \Lambda = (\gamma, x, t)] = \mathbb{E}[\varphi_\gamma(S_{\gamma,x,t})],$$

see [45, Chapter 4] and [1, Theorem 13.46]. Combined with (15) and (16), this proves the claim. □

Next, we establish the stability of the statement in Theorem A.1 w.r.t. approximate data generation via the Euler-Maruyama scheme.

Theorem A.2 (Approximated Learning Problem)**.** *Let Assumptions A.1 and A.2 be satisfied and for every $M \in \mathbb{N}$ let*

$$\bar{u}^M \colon D \times [v, w]^d \times [0, T] \to \mathbb{R}$$

be the (up to sets of Lebesgue measure zero) unique solution of the approximated learning problem

$$\min_f \mathbb{E}\Big[\big(f(\Lambda) - Y^M\big)^2\Big]$$

where the minimum is taken over all measurable functions $f \colon D \times [v, w]^d \times [0, T] \to \mathbb{R}$. Then there exists a constant $C > 0$ such that for every $M \in \mathbb{N}$ it holds that

$$\|\bar{u}^M - \bar{u}\|_{\mathcal{L}^\infty(D \times [v,w]^d \times [0,T])} \leq \tfrac{C}{\sqrt{M}}.$$

Proof. Extending results on the Euler-Maruyama scheme (see, e.g., [34, Theorem 10.2.2]) one can prove that also in the parametric case for every $p \geq 2$ there exists a constant $C > 0$ such that for every $M \in \mathbb{N}$, $(\gamma, x, t) \in D \times [v, w]^d \times [0, T]$ it holds that

$$\mathbb{E}\big[\|S^{M,M}_{\gamma,x,t}\|^p\big] \leq C \quad \text{and} \quad \big(\mathbb{E}\big[\|S^{M,M}_{\gamma,x,t} - S_{\gamma,x,t}\|^p\big]\big)^{1/p} \leq \tfrac{C}{\sqrt{M}}. \tag{17}$$

Similar to the proof of Theorem A.1 one can further establish that for every $M \in \mathbb{N}$ and almost every $(\gamma, x, t) \in D \times [v, w]^d \times [0, T]$ it holds that

$$\bar{u}^M(\gamma, x, t) = \mathbb{E}[\,Y^M |\, \Lambda = (\gamma, x, t)] = \mathbb{E}[\varphi_\Gamma(S^{M,M}_\Lambda) \mid \Lambda = (\gamma, x, t)] = \mathbb{E}[\varphi_\gamma(S^{M,M}_{\gamma,x,t})]$$

where the existence of functions Υ^M with analogous properties as in Assumptions A.2 is guaranteed by the Euler-Maruyama scheme. The local Lipschitz property of φ_γ now ensures that for every $M \in \mathbb{N}$ and almost every $(\gamma, x, t) \in D \times [v, w]^d \times [0, T]$ it holds that

$$\begin{aligned}|\bar{u}^M(\gamma, x, t) - \bar{u}(\gamma, x, t)| &= \big|\mathbb{E}\big[\varphi_\gamma(S^{M,M}_{\gamma,x,t})\big] - \mathbb{E}[\varphi_\gamma(S_{\gamma,x,t})]\big| \\ &\leq\ c\, \mathbb{E}\big[\|S^{M,M}_{\gamma,x,t} - S_{\gamma,x,t}\|\big(1 + \|S^{M,M}_{\gamma,x,t}\|^c + \|S_{\gamma,x,t}\|^c\big)\big]\end{aligned} \tag{18}$$

which together with the Cauchy-Schwarz inequality and (17) proves the theorem. □

Note that this result can also be used to show that our generalization result in Theorem 4 is not compromised by using data simulated by the Euler-Maruyama scheme.

Now we outline how to prove the simultaneous approximation of the parametric solution map and its partial derivatives by a neural networks without curse of dimensionality, i.e. with the network size scaling only polynomially in the underlying spatial dimension. In mathematical terms, we prove approximation results in the Sobolev norm $\|\cdot\|_{W^{1,\infty}}$, see [15]. As a motivating example, we take the heat equation from Section 3.3 and from now on we only consider feed-forward neural networks with ReLU activation function (ReLU networks), see e.g. [43, Section 2] for a precise definition.

Theorem A.3 (Sobolev Approximation)**.** *Let $a \in \mathbb{R}$, $b \in (a, \infty)$, and for every $d \in \mathbb{N}$ let*

$$\bar{u}_d(\gamma_\sigma, x, t) = \|x\|^2 + t \operatorname{Trace}(\gamma_\sigma \gamma_\sigma^*), \quad (\gamma_\sigma, x, t) \in [a, b]^{d \times d} \times [v, w]^d \times [0, T],$$

be the parametric solution map for the d-dimensional heat equation with paraboloid initial condition. Then there exists a constant $C > 0$ with the following property: For every $\varepsilon \in (0, 1/2)$, $d \in \mathbb{N}$ there exists a ReLU network $\Phi_{\varepsilon,d}$ with at most $\lfloor Cd^4 \log(d/\varepsilon) \rfloor$ parameters satisfying that

$$\|\Phi_{\varepsilon,d} - \bar{u}_d\|_{W^{1,\infty}([a,b]^{d\times d} \times [v,w]^d \times [0,T])} \leq \varepsilon.$$

Proof. Our result is based on ReLU network approximation results in [22, Propositions C.1 and C.2] and [21, Propositions III.2 and III.4], which are extensions of the work by Yarotsky [58]. Specifically, let $\Delta > 0$ and let $\mathrm{sq}\colon [-\Delta, \Delta] \to \mathbb{R}$ be the squaring function given by $\mathrm{sq}(x) := x^2$. Then there exists a ReLU network Φ_ε^{sq} with $\mathcal{O}(\log(1/\varepsilon))$ layers, $\mathcal{O}(1)$ neurons per layer, and parameters bounded by $\mathcal{O}(1)$ satisfying that

$$\|\Phi_\varepsilon^{sq} - \mathrm{sq}\,\|_{W^{1,\infty}([-\Delta,\Delta])} \leq \varepsilon.$$

By the polarization identity $xy = \frac{1}{2}((x+y)^2 - x^2 - y^2)$ an analogous result holds for the multiplication function $\mathrm{mult}\colon [-\Delta, \Delta]^2 \to \mathbb{R}$ given by $\mathrm{mult}(x, y) := xy$. We can therefore imitate the representation

$$\bar{u}_d(\gamma_\sigma, x, t) = \sum_{i=1}^d \mathrm{sq}(x_i) + \sum_{i,j=1}^d \mathrm{mult}\left(t, \mathrm{sq}((\gamma_\sigma)_{ij})\right)$$

using ReLU network concatenation and parallelization [14, Section 5]. Finally, we can estimate the error using a chain rule for ReLU networks, see [7] and [22, Section B.1]. □

Next, we show that our setting even allows for combined approximation and generalization results without curse of dimensionality. To prove this, we focus on the d-dimensional heat equation with varying diffusivity and Gaussian initial condition. We first show that ReLU networks are capable of efficiently approximating the parametric solution map.

Theorem A.4 (Approximation)**.** *Let $a \in \mathbb{R}$, $b \in (a, \infty)$ and for every $d \in \mathbb{N}$ let*

$$\bar{u}_d(\gamma_\sigma, x, t) = \frac{1}{(1 + 2t\gamma_\sigma^2)^{d/2}} e^{-\frac{\|x\|^2}{1+2t\gamma_\sigma^2}}, \quad (\gamma_\sigma, x, t) \in [a, b] \times [v, w]^d \times [0, T], \tag{19}$$

be the parametric solution map of the d-dimensional heat equation with Gaussian initial condition. Then there exist a constant $C > 0$ and a polynomial $q\colon \mathbb{R} \to \mathbb{R}$ with the following property: For every $\varepsilon \in (0, 1/2)$, $d \in \mathbb{N}$ there exists a ReLU network $\Phi_{\varepsilon,d}$ with at most $\lfloor q(\log(d/\varepsilon)) \rfloor$ layers, at most $\lfloor Cd \rfloor$ neurons per layer, and parameters bounded by C satisfying that

$$\|\Phi_{\varepsilon,d} - \bar{u}_d\|_{\mathcal{L}^\infty([a,b] \times [v,w]^d \times [0,T])} \leq \varepsilon.$$

Proof. The proof is based on combining ReLU approximation results for Chebyshev polynomials (see [21, Lemma A.6]) and the squaring and multiplication functions sq, mult (see the proof of Theorem A.3). Specifically, for given $\Delta > 0$ we can approximate the functions

$$[0, \Delta] \ni x \mapsto h(x) := \sqrt{\tfrac{1}{1+2x}} \quad \text{and} \quad [0, \Delta] \ni x \mapsto g(x) := e^{-x^2}$$

up to precision ε by ReLU networks with $\mathcal{O}(\mathrm{polylog}(1/\varepsilon))$ layers, $\mathcal{O}(1)$ neurons per layer, and parameters bounded by $\mathcal{O}(1)$. Moreover, observe that

$$\bar{u}_d(\gamma_\sigma, x, t) = \prod_{i=1}^d \mathrm{mult}\left(g\big(\mathrm{mult}(x_i, f(t, \gamma_\sigma))\big), f(t, \gamma_\sigma)\right)$$

where

$$f(t, \gamma_\sigma) := h(\mathrm{mult}(t, \mathrm{sq}(\gamma_\sigma))) = \sqrt{\tfrac{1}{1+2t\gamma_\sigma^2}}.$$

We can imitate this representation using ReLU network concatenation and parallelization and hierarchical, pairwise multiplications for the tensor product, see [14, Section 5 and Proposition 6.4]. Finally, we can estimate the error via the mean value theorem. □

Now we show that the number of samples s in (7), needed to learn the parametric solution map $\bar{u}$, does not suffer from the curse of dimensionality, either. To satisfy boundedness assumptions commonly used in statistical learning theory, we restrict ourself to clipped ReLU networks, the output of which is assumed to be bounded by 1. This can be achieved by composing each ReLU network with a simple clipping function, which itself can be represented as a small ReLU network [8, Section A.4]. Note that this incorporates our prior knowledge that the parametric solution map of the heat equation with Gaussian initial condition in (19) satisfies $\|\bar{u}_d\|_{\mathcal{L}^\infty([a,b]\times[v,w]^d\times[0,T])} \leq 1$.

Theorem A.5 (Generalization)**.** *Let $a \in \mathbb{R}$, $b \in (a, \infty)$ and for every $d \in \mathbb{N}$ let*

$$V_d := \mathrm{vol}([a,b]\times[v,w]^d\times[0,T]) = T(b-a)(w-v)^d,$$

let $\bar{u}_d \colon [a,b]\times[v,w]^d\times[0,T] \mapsto [0,1]$ be the parametric solution map of the d-dimensional heat equation with Gaussian initial condition as defined in (19)*, let*

$$\Lambda_d = (\Gamma_d, X_d, \mathcal{T}_d) \sim \mathcal{U}([a,b]\times[v,w]^d\times[0,T]) \quad \textit{and} \quad N_d \sim \mathcal{N}(0, I_d)$$

be independent random variables, define $Y_d = e^{-\|X_d + \sqrt{\mathcal{T}_d}\,\Gamma_d N_d\|^2}$, and let $((\Lambda_{d,i}, Y_{d,i}))_{i\in\mathbb{N}}$ be i.i.d. random variables with $(\Lambda_{d,1}, Y_{d,1}) \sim (\Lambda_d, Y_d)$. Then there exist a constant $C > 0$ and a polynomial $q \colon \mathbb{R} \to \mathbb{R}$ with the following property: For every $\varepsilon, \rho \in (0, 1/2)$, $d, s \in \mathbb{N}$ with

$$s \geq (d/\varepsilon)^2 q(\log(d/\varepsilon)) \log(1/\rho)$$

there exists a neural network architecture $\mathcal{A}_{\varepsilon,d}$ with at most $\lfloor q(\log(d/\varepsilon))\rfloor$ layers and at most $\lfloor Cd\rfloor$ neurons per layer such that every measurable empirical risk minimizer

$$\hat{\Phi}_{\varepsilon,d,s} \colon \Omega \to \mathcal{H}_{\varepsilon,d}, \quad \hat{\Phi}_{\varepsilon,d,s}(\omega) \in \operatorname*{arg\,min}_{\Phi\in\mathcal{H}_{\varepsilon,d}} \tfrac{1}{s}\sum_{i=1}^{s} (\Phi(\Lambda_{d,i}(\omega)) - Y_{d,i}(\omega))^2, \quad \omega \in \Omega,$$

in a hypothesis space $\mathcal{H}_{\varepsilon,d}$ of clipped ReLU networks with architecture $\mathcal{A}_{\varepsilon,d}$ and parameters bounded by C satisfies that

$$\mathbb{P}\left[\tfrac{1}{V_d}\|\hat{\Phi}_{\varepsilon,d,s} - \bar{u}_d\|^2_{\mathcal{L}^2([a,b]\times[v,w]^d\times[0,T])} \leq \varepsilon\right] \geq 1 - \rho.$$

Proof. Let $\mathcal{A}_{\varepsilon,d}$ be the architecture of the ReLU network $\Phi_{\varepsilon/2,d}$ in Theorem A.4. To simplify notation, we define $\|\cdot\|_{\mathcal{L}^2} := \|\cdot\|_{\mathcal{L}^2([a,b]\times[v,w]^d\times[0,T])}$ and for every $\Phi \in \mathcal{H}_{\varepsilon,d}$ we define its risk $\mathcal{R}(\Phi)$ and its empirical risk $\hat{\mathcal{R}}(\Phi)$ by

$$\mathcal{R}(\Phi) := \mathbb{E}\Big[\big(\Phi(\Lambda_d) - Y_d\big)^2\Big] \quad \text{and} \quad \hat{\mathcal{R}}(\Phi) := \tfrac{1}{s}\sum_{i=1}^{s}(\Phi(\Lambda_{d,i}) - Y_{d,i})^2.$$

The fact that the regression function coincides with the parametric solution map (see Theorem A.1) and the bias-variance decomposition (see [8, Lemma 2.2] and [11]) imply that

$$\tfrac{1}{V_d}\|\hat{\Phi}_{\varepsilon,d,s} - \bar{u}_d\|^2_{\mathcal{L}^2} = \underbrace{\mathcal{R}(\hat{\Phi}_{\varepsilon,d,s}) - \mathcal{R}(\Phi^*)}_{\text{generalization error}} + \underbrace{\tfrac{1}{V_d}\|\Phi^* - \bar{u}_d\|^2_{\mathcal{L}^2}}_{\text{approximation error}}$$

where $\Phi^* \in \arg\min_{\Phi\in\mathcal{H}_{\varepsilon,d}} \|\Phi - \bar{u}_d\|_{\mathcal{L}^2}$ is a best approximation of $\bar{u}_d$ in $\mathcal{H}_{\varepsilon,d}$. Our choice of $\mathcal{A}_{\varepsilon,d}$ and Theorem A.4 ensure that

$$\tfrac{1}{V_d}\|\Phi^* - \bar{u}_d\|^2_{\mathcal{L}^2} \leq \|\Phi^* - \bar{u}_d\|^2_{\mathcal{L}^\infty([a,b]\times[v,w]^d\times[0,T])} \leq \varepsilon/2.$$

For the generalization error we make use of results on the covering numbers of neural network hypothesis spaces, see e.g. [8, Proposition 2.8]. They ensure the existence of clipped ReLU networks $(\Phi_i)_{i=1}^n \subset \mathcal{H}_{\varepsilon,d}$ with

$$\log(n) \in \mathcal{O}(d^2\,\mathrm{polylog}(d/\varepsilon)) \tag{20}$$

such that balls of radius $\varepsilon/64$ (w.r.t. the uniform norm) around those functions cover $\mathcal{H}_{\varepsilon,d}$. Further, note that the boundedness of the target variable, i.e. $\sup_{\omega\in\Omega}|Y_d(\omega)| \leq 1$, and the boundedness of the clipped ReLU networks in our hypothesis space, i.e. $\sup_{\Phi\in\mathcal{H}_{\varepsilon,d}}\|\Phi\|_{\mathcal{L}^\infty([a,b]\times[v,w]^d\times[0,T])} \leq 1$, ensure that the (empirical) risk is (uniformly) Lipschitz continuous with

$$\mathrm{Lip}(\mathcal{R}) \leq 4 \quad \text{and} \quad \mathrm{Lip}(\hat{\mathcal{R}}) \leq 4,$$

see [8, Proof of Theorem 2.4]. Thus we can bound the generalization error by

$$\begin{aligned}\mathcal{R}(\hat{\Phi}_{\varepsilon,d,s}) - \mathcal{R}(\Phi^*) &\leq \mathcal{R}(\hat{\Phi}_{\varepsilon,d,s}) - \hat{\mathcal{R}}(\hat{\Phi}_{\varepsilon,d,s}) + \hat{\mathcal{R}}(\Phi^*) - \mathcal{R}(\Phi^*) \\ &\leq 2 \max_{i=1}^{n} \left|\mathcal{R}(\Phi_i) - \hat{\mathcal{R}}(\Phi_i)\right| + \tfrac{2\varepsilon(\mathrm{Lip}(\mathcal{R})+\mathrm{Lip}(\hat{\mathcal{R}}))}{64} \\ &\leq 2 \max_{i=1}^{n} \left|\mathcal{R}(\Phi_i) - \hat{\mathcal{R}}(\Phi_i)\right| + \varepsilon/4.\end{aligned}$$

Employing Hoeffding's inequality [28] and a union bound, it holds that

$$\mathbb{P}\big[\max_{i=1}^{n} \left|\mathcal{R}(\Phi_i) - \hat{\mathcal{R}}(\Phi_i)\right| \leq \varepsilon/8\big] \geq 1 - \rho$$

where we need $s \sim \log(n/\rho)/\varepsilon^2$ many samples. Together with (20) this implies the claim. □

A.2 Implementation Details

First, we want to present a rigorous definition of our Multilevel network architecture.

Definition A.2 (Multilevel Architecture)**.** *Let $L, q, p \in \mathbb{N}$, $\chi \in \{0, 1\}$, and $\varrho\colon \mathbb{R} \to \mathbb{R}$. We define the Multilevel network $\Phi\colon \mathbb{R}^p \to \mathbb{R}$ with input dimension $\dim_{in}(\Phi) = p$, L levels, amplifying factor q, (component-wise applied) activation function ϱ, and residual constant χ for every $x \in \mathbb{R}^p$ by*

$$\Phi(x) := \sum_{l=0}^{L-1} \Phi_l^{2^l}(x) \in \mathbb{R} \tag{21}$$

where the intermediate network outputs $\Phi_l^i(x)$ are given by the following scheme:

$$\begin{aligned}\Phi_l^1(x) &= \mathcal{A}_l^1(\varrho(\mathrm{Norm}_l^1(\mathcal{A}_l^0(x))), && l \in \{0, \dots, L-1\}, \\ \Phi_l^i(x) &= \mathcal{A}_l^i(\varrho\,\mathrm{Norm}_l^i(\Phi_l^{i-1}(x) + \chi\Phi_{l+1}^{2i-2}(x))), && l \in \{1, \dots, L-2\},\ i \in \{2, \dots, 2^l\}, \\ \Phi_{L-1}^i(x) &= \mathcal{A}_{L-1}^i(\varrho(\mathrm{Norm}_{L-1}^i(\Phi_{L-1}^{i-1}(x))), && i \in \{2, \dots, 2^{L-1}\}.\end{aligned}$$

In the above, the constant χ controls whether we use intermediate residual connections, and for every $l \in \{0, \dots, L-1\}$ the functions

$$\mathrm{Norm}_l^i\colon \mathbb{R}^{qp} \to \mathbb{R}^{qp}, \quad i \in \{1, \dots, 2^l\},$$

are denoting normalization layers, e.g. batch normalization [30] or layer normalization [3], and

$$\mathcal{A}_l^0\colon \mathbb{R}^p \to \mathbb{R}^{qp}, \quad \mathcal{A}_l^i\colon \mathbb{R}^{qp} \to \mathbb{R}^{qp}, \quad i \in \{1, \dots, 2^l - 1\}, \quad \mathcal{A}_l^{2^l}\colon \mathbb{R}^{qp} \to \mathbb{R}$$

are learnable linear mappings (or affine-linear in case of $\mathcal{A}_l^{2^l}$).

In the implementation of our examples we used $\chi = 1$ to propagate intermediate residuals from the corresponding higher level using additive skip-connections, followed by a batch normalization layer as proposed by [30]. This allows the length of the shortest gradient path during backpropagation to scale like the number of levels L instead of the number of layers 2^L; a feature commonly known to prevent diminishing or exploding gradients [59]. Thus, we can maintain computational tractability while at the same time having rather deep architectures. Note that a certain depth is needed for our approximation and generalization results in Section A.1, as well as to optimally approximate certain families of functions [40, 43, 58]. We pick the ReLU activation function as non-linearity to remain consistent with our theoretical guarantees in Section A.1 and with the growing body of literature on the approximation and generalization capabilities of ReLU networks. To optimize the networks we use the Adam optimizer (with decoupled weight decay regularization as proposed by [39]) and exponentially decaying learning rate. The precise setup is summarized in Table 5 and the hyperparameters over which we optimized using Tune [37, 38] are given in Table 6.

Table 5: Training setup

	Black-Scholes	**Basket Put**	**Heat Paraboloid**	**Heat Gaussian**
Input sets				
D_σ	$[0.1, 0.6] \times \{0\}$	$([0.1, 0.6]^{3\times 3})^4$	$\{\vec{0}\} \times [0, 1]^{10\times 10}$	$\{\vec{0}\} \times [0, 0.1]I_{150}$
D_μ	$\{\vec{0}\}$	$[0.1, 0.6]^{3\times 4}$	$\{\vec{0}\}$	$\{\vec{0}\}$
D_φ	$[10, 12]$	$[10, 12]$	$\{\}$	$\{\}$
$[v, w]$	$[9, 10]$	$[9, 10]$	$[0.5, 1.5]$	$[-0.1, 0.1]$
$[0, T]$	$[0, 1]$	$[0, 1]$	$[0, 1]$	$[0, 1]$
Network				
$\dim_{\text{in}}(\Phi)$	4	53	111	152
architecture	Multilevel	Multilevel	Multilevel	Multilevel
(L, q, χ)	(4,5,1)	(4,5,1)	(4,4,1)	(4,4,1)
activation ϱ	ReLU	ReLU	ReLU	ReLU
Norm layer	batch norm.	batch norm.	batch norm.	batch norm.
#parameters	5.4K	0.8M	2.4M	4.5M
Training				
solution SDE	analytic	Euler-M.	analytic	analytic
optimizer	AdamW	AdamW	AdamW	AdamW
param. init.	$\mathcal{U}([-\xi, \xi])$	$\mathcal{U}([-\xi, \xi])$	$\mathcal{U}([-\xi, \xi])$	$\mathcal{U}([-\xi, \xi])$
weight decay	0.01	0.01	0.01	0.01
batch-size	2^{16}	2^{17}	2^{17}	2^{17}
(init. lr., min. lr.)	$(10^{-2}, 10^{-8})$	$(10^{-3}, 10^{-8})$	$(10^{-3}, 10^{-8})$	$(10^{-3}, 10^{-8})$
(decay, patience)	$(0.25, 4000)$	$(0.4, 4000)$	$(0.4, 4000)$	$(0.4, 4000)$
Validation				
solution PDE	analytic	MC-approx.	analytic	analytic
batch-size	2^{16}	2^{17}	2^{17}	2^{17}
#eval. batches	150	1	150	150
Execution				
seeds	0,1,2,3	0,1,2,3	0,1,2,3	0,1,2,3
#GPUs per run	2 (Tesla V100)	4 (Tesla V100)	2 (Tesla V100)	2 (Tesla V100)

1. **Input sets:** input sets for the parameter $\gamma = (\gamma_\sigma, \gamma_\mu, \gamma_\varphi) \in D_\sigma \times D_\mu \times D_\varphi = D$, the spatial variable $x \in [v, w]$, and the time variable $t \in [0, T]$, as defined in Section 2.1.
2. **Network:** input dimension $\dim_{\text{in}}(\Phi)$, activation function ϱ, number of levels L, amplifying factor q, usage of intermediate residual connections χ, normalization layers Norm_l^i, and approximate number of parameters of the Multilevel architecture, see Definition A.2.
3. **Training:** computation of the SDE solution, optimizer, initialization of the linear mappings $\mathcal{A}_l^i$ where $\xi := d_{\text{in}}^{-1/2}$ with d_{in} denoting the input dimension, weight decay, batch-size, initial learning rate, and factor for learning rate decay each patience steps as long as the learning rate is larger than the minimal learning rate. Note that the training data size in (7) is given by $s =$ batch-size $\cdot$ #steps where the number of steps is reported in Tables 1, 2, 3, and 4.
4. **Validation:** pointwise computation of the PDE solution, batch-size, and number of batches per evaluation.[10] Note that $n =$ batch-size $\cdot$ #eval. batches for each reported $\mathcal{L}^1$-error, see (9).
5. **Execution:** PyTorch module and random module seeds for the 4 independent runs and number and type of GPUs per run.

[10]The evaluation of the PDE via Monte Carlo simulation as in (10) is computationally very expensive. That is the reason why we only took one evaluation batch per iteration for the Basket put option. However, note that training the network with Euler-Maruyama simulated data does not increase the training time significantly (see Table 2) which underlines the general applicability of our algorithm.

Table 6: Ranges for hyperparameter optimization

hyperparameter	range
(L, q)	$\{3, 4\} \times \{4, 5, 6\}$
optimizer	{AdamW, SGD (with momentum & weight decay)}
batch-size	$\{16384, 32768, 65536, 131072\}$
learning rate	$(10^{-1}, 10^{-5})$
lr. decay factor	$(0.2, 0.6)$

Table 7: Ablation study for the Black-Scholes model

architecture, normalization layer	avg. time (s)	avg. best $\mathcal{L}^1$-error	#parameters
Feed-Forward, layer norm.	809 ± 9	0.1476 ± 0.0772	6741
Feed-Forward, none	496 ± 26	0.0526 ± 0.0002	6101
Feed-Forward, batch norm.	3755 ± 57	0.0017 ± 0.0003	6741
Multilevel $\chi = 0$, layer norm.	867 ± 10	0.0349 ± 0.0000	5404
Multilevel $\chi = 0$, none	570 ± 6	0.0069 ± 0.0001	4804
Multilevel $\chi = 0$, batch norm.	3414 ± 18	0.0012 ± 0.0000	5404
Multilevel $\chi = 1$, layer norm.	874 ± 13	0.0348 ± 0.0001	5404
Multilevel $\chi = 1$, none	581 ± 10	0.0069 ± 0.0000	4804
Multilevel $\chi = 1$, batch norm.	3453 ± 34	$\mathbf{0.0011} \pm 0.0001$	5404

Table 8: Ablation study for the heat equation with paraboloid initial condition

architecture	avg. time (s)	avg. best $\mathcal{L}^1$-error	#parameters
Feed-Forward	14764 ± 65	0.0090 ± 0.0003	3020977
Multilevel $\chi = 0$	13892 ± 83	0.0058 ± 0.0001	2380732
Multilevel $\chi = 1$	14049 ± 138	$\mathbf{0.0055} \pm 0.0001$	2380732

A.3 Additional Numerical Results

In Tables 7 and 8 we present an ablation study which empirically proves the superior performance of our Multilevel architecture in combination with batch normalization compared to feed-forward architectures or the usage of layer normalization [3]. For the feed-forward architecture we used the network $\Phi_L^{2^L}$ defined in (21), i.e. only the highest level of the corresponding Multilevel network with $L + 1$ levels and $\chi = 0$. Despite having slightly less parameters, our Multilevel architecture consistently outperforms the feed-forward architecture. Moreover, the use of residual connections, i.e. $\chi = 1$, has a positive impact. Note that all not-mentioned settings are kept as in Table 5.

The performance of our algorithm in the case of the Black-Scholes option pricing model from Section 3.1 is further illustrated in Figures 5, 6, 7, and 8. Finally, Figure 9 depicts the computational cost of our algorithm as a function of the problem input dimension for the heat equation with paraboloid initial condition.

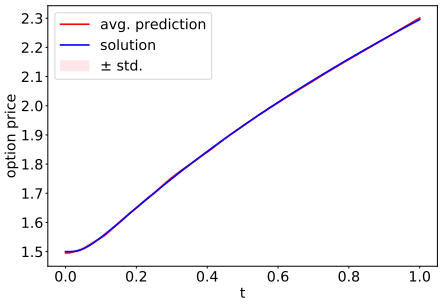


Figure 5: Shows $\bar{u}(\gamma, x, \cdot)$ vs. the average prediction (and its standard deviation) at $x = 9.5$, $\gamma_\sigma = 0.35$, and $\gamma_\varphi = 11$.

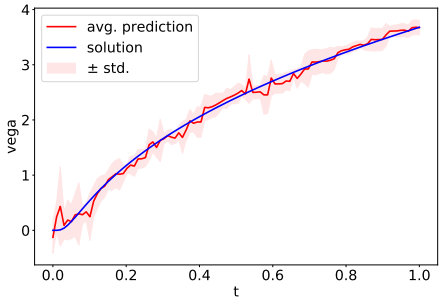


Figure 6: Shows the Vega $\frac{\partial \bar{u}}{\partial \gamma_\sigma}(\gamma, x, \cdot)$ vs. the average prediction (and its standard deviation) at $x = 9.5$, $\gamma_\sigma = 0.35$, and $\gamma_\varphi = 11$.

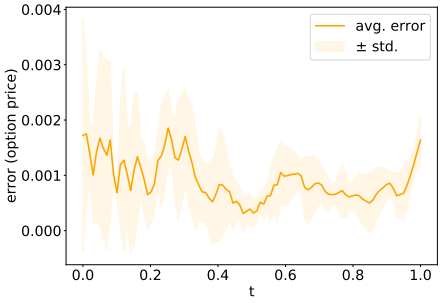


Figure 7: Shows the average prediction error $\frac{|\Phi(\gamma,x,\cdot)-\bar{u}(\gamma,x,\cdot)|}{1+|\bar{u}(\gamma,x,\cdot)|}$ and its standard deviation at $x = 9.5$, $\gamma_\sigma = 0.35$, and $\gamma_\varphi = 11$.

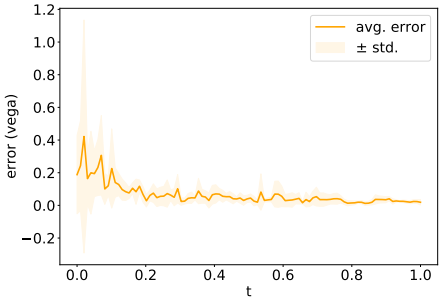


Figure 8: Shows the average error of the Vega $\frac{|\frac{\partial \Phi}{\partial \gamma_\sigma}(\gamma,x,\cdot)-\frac{\partial \bar{u}}{\partial \gamma_\sigma}(\gamma,x,\cdot)|}{1+|\frac{\partial \bar{u}}{\partial \gamma_\sigma}(\gamma,x,\cdot)|}$ and its standard deviation at $x = 9.5$, $\gamma_\sigma = 0.35$, and $\gamma_\varphi = 11$.

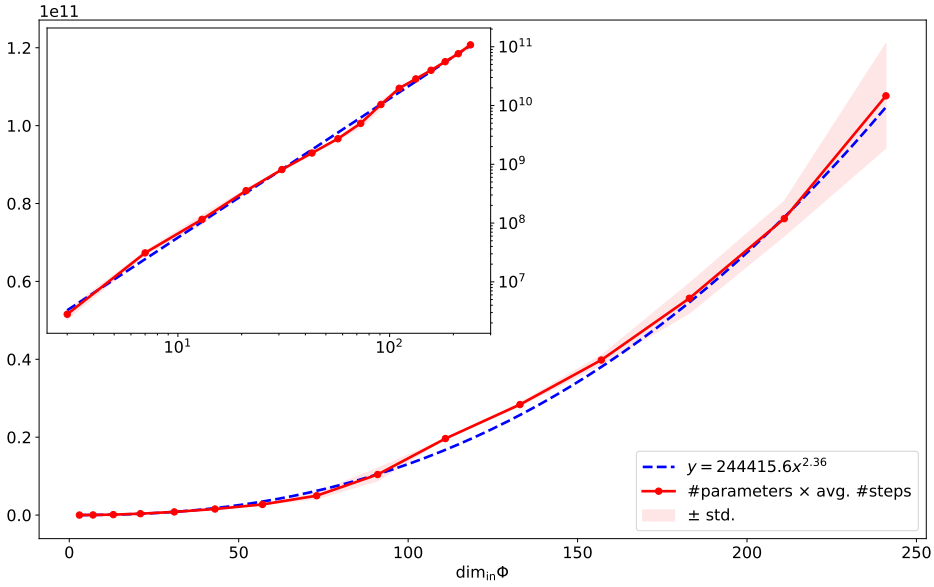


Figure 9: Shows the cost in terms of number of network parameters times average number of steps to achieve an $\mathcal{L}^1$-error of 10^{-2} w.r.t. to the problem dimension $d^2 + d + 1$ for the heat equations with paraboloid initial condition and $d = 1, \ldots, 15$. The absence of the curse of dimensionality is underlined by the linear behaviour in the log-log inset. The error was evaluated every 250 gradient descent steps and except of the varying dimension all settings are kept as in Table 5.

V Robust SDE-Based Variational Formulations for Solving Linear PDEs via Deep Learning

Comments

Conference: Spotlight and poster presentation at ICML 2022.
Code: github.com/juliusberner/robust_kolmogorov
E-Print: arXiv:2206.10588[cs.LG]

Contribution

Lorenz Richter and Julius Berner contributed equally to the ideas, development, and writing of the paper. The numerical experiments have been implemented by Julius Berner.

Bibliographic Information

Richter, Lorenz and Julius Berner (2022). "Robust SDE-Based Variational Formulations for Solving Linear PDEs via Deep Learning." In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 18649–18666.

Robust SDE-Based Variational Formulations for Solving Linear PDEs via Deep Learning

Lorenz Richter [*1 2 3] **Julius Berner** [*4]

Abstract

The combination of Monte Carlo methods and deep learning has recently led to efficient algorithms for solving partial differential equations (PDEs) in high dimensions. Related learning problems are often stated as variational formulations based on associated stochastic differential equations (SDEs), which allow the minimization of corresponding losses using gradient-based optimization methods. In respective numerical implementations it is therefore crucial to rely on adequate gradient estimators that exhibit low variance in order to reach convergence accurately and swiftly. In this article, we rigorously investigate corresponding numerical aspects that appear in the context of linear Kolmogorov PDEs. In particular, we systematically compare existing deep learning approaches and provide theoretical explanations for their performances. Subsequently, we suggest novel methods that can be shown to be more robust both theoretically and numerically, leading to substantial performance improvements.

1. Introduction

In this paper we suggest novel methods for solving high-dimensional linear Kolmogorov PDEs. In particular, those methods allow for more efficient approximations by reducing the variance of loss estimators, as illustrated by numerical experiments like the one displayed in Figure 1.

High-dimensional PDEs are ubiquitous in applications appearing in economics, science, and engineering, however, their numerical treatment poses formidable challenges since traditional grid-based methods suffer from the curse of dimensionality. Recently, the combination of Monte Carlo methods and deep learning has led to feasible algorithms (E et al., 2017; Sirignano & Spiliopoulos, 2018; Han et al., 2018; Raissi et al., 2019; Pfau et al., 2020; Berner et al., 2021), which, at least in some settings, are able to beat this curse (Jentzen et al., 2018; Reisinger & Zhang, 2019; Berner et al., 2020b). The general idea is to rely on stochastic representations of the PDE which may be interpreted as variational formulations that can subsequently be minimized by iterative gradient decent methods (Nüsken & Richter, 2021a). In respective algorithms it is therefore crucial to rely on adequate gradient estimators that exhibit low variance and eventually lead to robust numerical optimization routines, where robustness refers to the ability of coping with stochasticity in the optimization process.

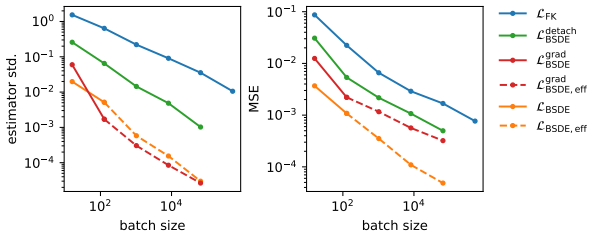


Figure 1. Standard deviation and performance (in terms of the mean squared error (MSE) on evaluation data) of different losses as a function of the batch size after training a neural network for $30k$ steps, where due to memory constraints different losses allow for different maximal batch size values.

In this article we focus on linear Kolmogorov PDEs for which variational formulations may be based on the celebrated Feynman-Kac formula (Beck et al., 2021). This method is handy from an implementational point of view, however, we can demonstrate potential non-robustness issues that might deteriorate the numerical performance. To overcome these issues, we suggest alternative approaches resting on backward stochastic differential equations which lead to variance-reduced estimators and robust algorithms. Our contributions can be summarized as follows:

- We provide a systematic numerical comparison of loss

[*]Equal contribution (the author order was determined by `numpy.random.rand(1)`) [1]dida Datenschmiede GmbH, Germany [2]Zuse Institute Berlin, Germany [3]Institute of Mathematics, Freie Universität Berlin, Germany [4]Faculty of Mathematics, University of Vienna, Austria. Correspondence to: Julius Berner <julius.berner@univie.ac.at>, Lorenz Richter <lorenz.richter@fu-berlin.de>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

functions for solving linear Kolmogorov PDEs via deep learning.

- We can show both theoretically and numerically why certain methods are non-robust and might therefore incur numerical instabilities.
- We provide new loss functions and optimization methods which are provably robust and ultimately yield significant performance improvements in relevant high-dimensional numerical experiments.

1.1. Related Work

Solving linear Kolmogorov PDEs via the Feynman-Kac formula by means of deep learning has been suggested in Beck et al. (2021) and further analyzed in Berner et al. (2020a). For an analysis of certain kinds of elliptic linear PDEs we refer to Grohs & Herrmann (2020). Variational formulations of nonlinear PDEs based on backward stochastic differential equations (BSDEs) have been pioneered in E et al. (2017) and extensions that aim for approximations on entire domains have, for instance, been suggested in Nüsken & Richter (2021a). We refer to Richter (2021) for a comprehensive introduction and further perspectives, and to Beck et al. (2019) for the treatement of fully nonlinear equations. An alternative method for the approximation of high-dimensional PDEs is termed *physics-informed neural networks* (PINNs) (Raissi et al., 2019; Sirignano & Spiliopoulos, 2018), which relies on iterative residual minimizations. Note that in contrast to most SDE-based methods it relies on the explicit computation of Hessians and involves additional loss terms for the boundary values, which need to be tuned carefully.

In terms of variance reduction of Monte Carlo estimators we refer to Vidales et al. (2018) for a control variate attempt via deep learning. Robustness and variance analyses of certain loss functions related to PDE problems have been conducted in Nüsken & Richter (2021b). The related observation that including a certain Itô integral term into the objective function can lead to variance-reduced estimators has been made in Zhou et al. (2021). Both works, however, consider different settings and focus only on Hamilton–Jacobi–Bellman (HJB) equations. Furthermore, they do not provide a detailed analysis of the interplay between robustness properties, performance, and computational feasibility, which is crucial for practical applications.

1.2. Notation

We assume that our random variables are defined on a suitable underlying filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_t, \mathbb{P})$ satisfying the usual conditions (Klenke, 2013). We denote the expectation of a random variable $A\colon \Omega \to \mathbb{R}$ by $\mathbb{E}[A] \coloneqq \int A \,\mathrm{d}\mathbb{P}$. Given another random variable $B\colon \Omega \to \mathbb{R}^k$ with $k \in \mathbb{N}$, we also consider the conditional expectation $\mathbb{E}[A|B]$ of A given B and the conditional expectation $b \mapsto \mathbb{E}[A|B = b]$ of A given $B = b$, see, e.g., Ash & Doléans-Dade (2000) for the definitions. We further define the variance of A by $\mathbb{V}[A] \coloneqq \mathbb{E}\left[(A - \mathbb{E}[A])^2\right]$ and the conditional variance of A given B by $\mathbb{V}[A|B] \coloneqq \mathbb{E}\left[(A - \mathbb{E}[A|B])^2|B\right]$. Throughout the article, W is a standard d-dimensional Brownian motion which is adapted to the underlying filtration and we consider stochastic integrals w.r.t. W as, e.g., defined in Gall (2016). Further remarks on the notation can be found in Appendix A.

2. SDE-Based Variational Formulations of Linear PDEs

In this paper we consider linear *Kolmogorov partial differential equations*[1] of the type

$$(\partial_t + L)V(x,t) = 0, \qquad (x,t) \in \mathbb{R}^d \times [0,T), \quad \text{(1a)}$$

$$V(x,T) = g(x), \qquad x \in \mathbb{R}^d, \quad \text{(1b)}$$

where $g \in C(\mathbb{R}^d, \mathbb{R})$ is a given terminal condition and

$$L \coloneqq \frac{1}{2}\sum_{i,j=1}^{d} (\sigma\sigma^\top)_{ij}(x,t)\partial_{x_i}\partial_{x_j} + \sum_{i=1}^{d} b_i(x,t)\partial_{x_i} \quad (2)$$

is a differential operator based on given coefficient functions $\sigma \in C(\mathbb{R}^d \times [0,T], \mathbb{R}^{d\times d})$ and $b \in C(\mathbb{R}^d \times [0,T], \mathbb{R}^d)$. For simplicity, let us assume the existence of a unique strong solution $V \in C^{2,1}(\mathbb{R}^d \times [0,T], \mathbb{R})$ to the PDE in (1) and refer to Assumption 1 in Appendix B for further technical details. At the same time, let us note that most of our results also hold for the more general concept of viscosity solutions that are continuous but not necessarily differentiable and require weaker conditions on the functions b, σ, and g, see Hairer et al. (2015). Furthermore, we can also consider more general linear PDEs as well as elliptic and parabolic problems on bounded domains, as elaborated on in Appendix C.

This covers a broad spectrum of PDEs, for which accurate and reliable solvers are of great importance to practitioners. For instance, such PDEs frequently appear in physics for the modelling of heat flow and diffusion processes (Widder, 1976; Pascucci, 2005). Moreover, the PDE in (1) includes the Black-Scholes equation and extensions thereof, used for pricing financial derivative instruments (Black & Scholes, 1973; Pironneau & Achdou, 2009; Ekström & Tysk, 2010). Finally, we want to emphasize appearances of such PDEs in the field of machine learning, e.g., in the context of reinforcement learning (Theodorou et al., 2010; Kiumarsi et al., 2017) and diffusion-based generative modeling (Huang et al., 2021).

[1] PDEs of this type are often also referred to as *Kolmogorov (backward) equations* and their adjoints are known as *Fokker-Planck equations* or *Kolmogorov forward equations.*

In order to design machine learning algorithms aiming to approximate solutions to (1), we consider loss functions

$$\mathcal{L} : \mathcal{U} \to \mathbb{R}_{\geq 0}, \tag{3}$$

which shall be minimal if and only if $u \in \mathcal{U}$ fulfills (1), thereby offering a variational formulation that allows for iterative minimization strategies. Here, $\mathcal{U}$ is an appropriate function class holding sufficient approximation capacity and containing, for instance, a class of certain neural networks. Moreover, we implicitly assume that $\mathcal{U}$ is chosen such that $V \in \mathcal{U}$ and, for our theoretical results, we require the functions in $\mathcal{U}$ to be sufficiently smooth.

A first obvious choice for the loss could be the squared difference between the approximating function u and the solution V, i.e.

$$\mathcal{L}_{\text{Eval}}(u) \coloneqq \mathbb{E}\left[\left(V(\xi,\tau) - u(\xi,\tau)\right)^2\right], \tag{4}$$

where ξ and τ are suitable[2] random variables with values in $\mathbb{R}^d$ and $[0,T]$, respectively. At first glance, this loss seems intractable since the solution V is just the quantity we are after and is therefore not known. However, let us recall the *Feynman-Kac formula*, which connects the deterministic function V to a stochastic process by stating that almost surely it holds that

$$V(\xi,\tau) = \mathbb{E}\left[g(X_T)|(\xi,\tau)\right], \tag{5}$$

where X is the solution to the SDE

$$\mathrm{d}X_s = b(X_s,s)\,\mathrm{d}s + \sigma(X_s,s)\,\mathrm{d}W_s, \quad X_\tau = \xi, \tag{6}$$

see Appendix C for details. This allows us to rewrite (4) as

$$\begin{aligned}
\mathcal{L}_{\text{Eval}}(u) &= \mathbb{E}\left[\left(\mathbb{E}\left[g(X_T)|(\xi,\tau)\right] - u(\xi,\tau)\right)^2\right] && \text{(7a)}\\
&= \mathbb{E}\left[\mathbb{E}\left[\Delta_u|(\xi,\tau)\right]^2\right] && \text{(7b)}\\
&= \mathbb{E}\left[\mathbb{E}\left[\Delta_u - S_u|(\xi,\tau)\right]^2\right], && \text{(7c)}
\end{aligned}$$

which now relies on the (random) quantities[3]

$$\Delta_u \coloneqq g(X_T) - u(\xi,\tau), \tag{8a}$$

$$S_u \coloneqq \int_\tau^T \left(\sigma^\top \nabla_x u\right)(X_s,s)\cdot \mathrm{d}W_s. \tag{8b}$$

[2] In practice, one often chooses random variables that are uniformly distributed on a given hypercube. For our theoretical results, we assume that ξ and τ are $\mathcal{F}_0$-measurable and exhibit sufficiently many bounded moments. This ensures that ξ and τ are sufficiently concentrated and independent of the Brownian motion W.

[3] For the sake of readability, we omit several implicit dependencies. Specifically, the stochastic process X and the quantities Δ_u and S_u depend on the random variable (ξ,τ), defining the initial time and value of the SDE, i.e., $X_\tau = \xi$. Also note that the stochastic integral S_u and the SDE in (6) are driven by the same Brownian motion W.

The last step (7c) follows from the fact that, under mild regularity assumptions, the stochastic integral S_u is a martingale which has zero expectation given (ξ,τ). Adding quantities with a known expectation is a common trick for variance reduction of corresponding estimators known as *control variates*, see, e.g., Section 4.4.2 in Robert & Casella (2004). The benefit of particularly choosing S_u becomes clear with the following Lemma, which states that for $u = V$ the corresponding control variate actually yields a zero-variance estimator.

Lemma 2.1 (Optimal control variate)**.** *Let V be a solution to the PDE in (1). For Δ_V and S_V as defined in (8a) and (8b) it almost surely holds that*

$$\Delta_V = S_V, \tag{9}$$

which implies that $\mathbb{V}\left[\Delta_V - S_V|(\xi,\tau)\right] = 0$.

Proof. The proof is an application of Itô's lemma. Together with the observation that X is an Itô process given by the SDE in (6), it implies that almost surely it holds that

$$V(X_T,T) - V(\xi,\tau) = \int_\tau^T (\partial_t + L)V(X_s,s)\,\mathrm{d}s + S_V,$$

assuming that $V \in C^{2,1}(\mathbb{R}^d \times [0,T],\mathbb{R})$, see, e.g., Theorem 3.3.6 in Karatzas & Shreve (1998). Using the fact that V solves the PDE in (1), this implies the statement. □

While, in principle, the new representation in (7) makes the loss $\mathcal{L}_{\text{Eval}}$ computationally tractable, an immediate disadvantage are the two expectations, which, for a numerical implementation, might be costly as they would both need to rely on Monte Carlo approximations.

It turns out, however, that the conditional expectations in (7b) and (7c) are in fact not needed and we can further define the two losses

$$\mathcal{L}_{\text{FK}}(u) \coloneqq \mathbb{E}\left[\Delta_u^2\right] \tag{10}$$

and

$$\mathcal{L}_{\text{BSDE}}(u) \coloneqq \mathbb{E}\left[\left(\Delta_u - S_u\right)^2\right]. \tag{11}$$

Note that now the expectations in (10) and (11) are to be understood with respect to the random initial time τ, random initial value ξ, and the randomness of the Brownian motion W, which defines the evolution of the stochastic process X on the (random) interval $[\tau,T]$.

In the remainder of this article we will investigate the above losses with respect to certain robustness properties that will turn out to imply different statistical properties and consequently lead to different optimization performances of corresponding algorithms. Let us start with the following proposition which relates the three losses to one another and

therefore guarantees that $\mathcal{L}_{\mathrm{FK}}$ and $\mathcal{L}_{\mathrm{BSDE}}$ indeed constitute meaningful objectives for approximating the solution V to the PDE in (1).

Proposition 2.2 (Minima of $\mathcal{L}_{\mathrm{FK}}$ and $\mathcal{L}_{\mathrm{BSDE}}$)**.** *For every measurable* $u\colon \mathbb{R}^d \times [0,T] \to \mathbb{R}$ *it holds that*

$$\mathcal{L}_{\mathrm{Eval}}(u) = \mathcal{L}_{\mathrm{FK}}(u) - \mathbb{V}\left[S_V\right] \tag{12a}$$
$$= \mathcal{L}_{\mathrm{BSDE}}(u) - \mathbb{V}\left[S_{V-u}\right]. \tag{12b}$$

This implies that the solution V *to the PDE in* (1) *is the unique*[4] *minimizer of* $\mathcal{L}_{\mathrm{FK}}$ *and* $\mathcal{L}_{\mathrm{BSDE}}$*. The minima satisfy*

(i) $\min_u \mathcal{L}_{\mathrm{FK}}(u) = \mathcal{L}_{\mathrm{FK}}(V) = \mathbb{E}[\Delta_V^2] = \mathbb{V}[S_V]$,

(ii) $\min_u \mathcal{L}_{\mathrm{BSDE}}(u) = \mathcal{L}_{\mathrm{BSDE}}(V) = 0$.

Proof. See Appendix B. □

Remark 2.3 (Origin of the losses). As already hinted at, the names of the losses originate from the Feynman-Kac (FK) formula and backward stochastic differential equations (BSDE), respectively. The former, as stated in (5) and detailed in Appendix C, implies that the solution of the PDE in (1) can be expressed as a conditional expectation, i.e.,

$$V(x,t) = \mathbb{E}[g(X_T)|X_t = x] \tag{13a}$$
$$= \mathbb{E}[g(X_T)|(\xi,\tau) = (x,t)]. \tag{13b}$$

This can then be combined with the fact that the conditional expectation is the minimizer of a corresponding L^2-optimization problem and one readily recovers $\mathcal{L}_{\mathrm{FK}}$ as specified in (10), see Beck et al. (2021). The latter loss, $\mathcal{L}_{\mathrm{BSDE}}$, originates from the BSDE

$$\mathrm{d}Y_s = Z_s \cdot \mathrm{d}W_s, \quad Y_T = g(X_T), \tag{14}$$

which can essentially be derived from Itô's lemma and which involves the backward processes $Y_s = V(X_s, s)$ and $Z_s = (\sigma^\top \nabla_x V)(X_s, s)$, see Appendix D for further details. The name of the loss $\mathcal{L}_{\mathrm{Eval}}$ refers to the fact that it is mostly used in order to evaluate the solution candidate u, using either a known solution V or the reformulation in (7). This is also how we evaluate the mean squared error (MSE) for our methods, see Appendix E.

2.1. Estimator Versions of Losses and Robustness Issues

The expectations in the losses (4), (7), (10), or (11) can usually not be computed analytically and we need to resort to Monte Carlo approximations $\mathcal{L}^{(K)}$ based on a given number $K \in \mathbb{N}$ of independent samples

$$(\xi^{(k)}, \tau^{(k)}, W^{(k)}) \sim (\xi, \tau, W), \quad k = 1, \dots, K.$$

[4] Up to null sets w.r.t. to the image measure of (ξ, τ).

In practice, one further has to employ a time discretization of the SDE, yielding the estimators

$$\widehat{\mathcal{L}}^{(K)}_{\mathrm{FK}}(u) = \frac{1}{K}\sum_{k=1}^{K}\left(\widehat{\Delta}_u^{(k)}\right)^2, \tag{15a}$$

$$\widehat{\mathcal{L}}^{(K)}_{\mathrm{BSDE}}(u) = \frac{1}{K}\sum_{k=1}^{K}\left(\widehat{\Delta}_u^{(k)} - \widehat{S}_u^{(k)}\right)^2, \tag{15b}$$

where $\widehat{\Delta}_u^{(k)}$ and $\widehat{S}_u^{(k)}$ are discretized versions of (8a) and (8b), evaluated at the k-th sample. More precisely, we define

$$\widehat{\Delta}_u^{(k)} := g(\widehat{X}^{(k)}_{J^{(k)}+1}) - u(\xi^{(k)}, \tau^{(k)}), \tag{16a}$$

$$\widehat{S}_u^{(k)} := \sum_{j=1}^{J^{(k)}} s_j^{(k)}, \tag{16b}$$

where

$$s_j^{(k)} := (\sigma^\top \nabla_x u)(\widehat{X}_j^{(k)}, t_j^{(k)}) \cdot \left(W^{(k)}_{t_{j+1}} - W^{(k)}_{t_j}\right). \tag{17}$$

Here, $(\widehat{X}_j^{(k)})_{j=1}^{J^{(k)}+1}$ denotes a discretization of the solution to the SDE in (6) (driven by the Brownian motion $W^{(k)}$) on a time grid $\tau^{(k)} = t_1^{(k)} < \cdots < t^{(k)}_{J^{(k)}+1} = T$ with initial condition $\widehat{X}_1^{(k)} = \xi^{(k)}$, see Appendix E for details.

In this section we will investigate different statistical properties of the Monte Carlo estimators. It will turn out that even though the loss $\mathcal{L}_{\mathrm{FK}}$ is prominent for solving linear PDEs in practice (see, e.g., Beck et al. (2021); Berner et al. (2020a)), it might have some disadvantages from a numerical point of view. In fact, Lemma 2.1 and Proposition 2.2 already show that, while $\mathcal{L}_{\mathrm{BSDE}}$ can be minimized to zero expectation, $\mathcal{L}_{\mathrm{FK}}$ always exhibits the additional term

$$\mathbb{E}[\Delta_V^2] = \mathbb{V}[\Delta_V] = \mathbb{V}[S_V] = \mathbb{E}[S_V^2], \tag{18}$$

which is a potential source of additional noise. By the Itô isometry, this can also be written as

$$\mathbb{E}\left[S_V^2\right] = \mathbb{E}\left[\left(\int_\tau^T \|\sigma^\top \nabla_x V(X_s, s)\|^2 \mathrm{d}s\right)\right]. \tag{19}$$

The following proposition shows what this implies for the estimator versions of the losses at the optimum.

Proposition 2.4 (Variance of losses)**.** *For the estimator versions of the losses defined in* (10) *and* (11) *it holds that*

$$\mathbb{E}\left[\mathcal{L}^{(K)}_{\mathrm{FK}}(V)\right] = \mathbb{V}\left[S_V\right], \quad \mathbb{V}\left[\mathcal{L}^{(K)}_{\mathrm{FK}}(V)\right] = \frac{1}{K}\mathbb{V}\left[S_V^2\right]$$

and

$$\mathbb{E}\left[\mathcal{L}^{(K)}_{\mathrm{BSDE}}(V)\right] = 0, \qquad \mathbb{V}\left[\mathcal{L}^{(K)}_{\mathrm{BSDE}}(V)\right] = 0.$$

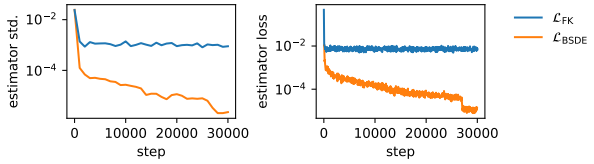


Figure 2. Estimator losses and their standard deviations as a function of the gradient steps when solving the HJB equation from Section 3.3 with batch size $K = 128$. One observes that the statement of Proposition 2.4 already holds in an approximative sense before converging to the solution V.

Proof. See Appendix B. □

Proposition 2.4 shows that neither the expectation nor the variance of $\mathcal{L}_{\mathrm{FK}}^{(K)}$ can be zero at the solution $u = V$ unless S_V or S_V^2 are deterministic, which is usually not the case in practice as demonstrated by the numerical examples in Section 3 and Figure 2. While the non-zero expectation of the estimator is typically not a problem for the optimization routine, the non-vanishing variance might cause troubles when converging to the global minimum.

Since the considered losses are usually minimized via variants of gradient decent, our further analysis shall therefore focus on the variance of corresponding gradient estimators. For this let us first define an appropriate notion of derivative.

Definition 2.5 (Gateaux derivative)**.** We say that $\mathcal{L}\colon \mathcal{U} \to \mathbb{R}_{\geq 0}$ is *Gateaux differentiable* at $u \in \mathcal{U}$ if for all $\phi \in \mathcal{U}$ the mapping

$$\varepsilon \mapsto \mathcal{L}(u + \varepsilon\phi) \tag{20}$$

is differentiable at $\varepsilon = 0$. The Gateaux derivative of $\mathcal{L}$ in direction ϕ is then defined as

$$\frac{\delta}{\delta u}\mathcal{L}(u;\phi) := \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0}\mathcal{L}(u + \varepsilon\phi). \tag{21}$$

Inspired by an analysis in Nüsken & Richter (2021b) let us first investigate the variance of derivatives at the solution $u = V$, where intuitively we might want to favor gradient estimators whose variances vanish at the global minimizer of the loss. The following proposition shows that this property is indeed only the case for $\mathcal{L}_{\mathrm{BSDE}}^{(K)}$.

Proposition 2.6 (Gradient variances)**.** *For the estimator versions of the losses defined in* (10) *and* (11) *and for all* $\phi \in \mathcal{U}$ *it holds that*

(i) $\mathbb{V}\left[\frac{\delta}{\delta u}\Big|_{u=V}\mathcal{L}_{\mathrm{FK}}^{(K)}(u;\phi)\right] = \frac{4}{K}\,\mathbb{V}\left[S_V\phi(\xi,\tau)\right],$

(ii) $\mathbb{V}\left[\frac{\delta}{\delta u}\Big|_{u=V}\mathcal{L}_{\mathrm{BSDE}}^{(K)}(u;\phi)\right] = 0.$

Proof. See Appendix B. □

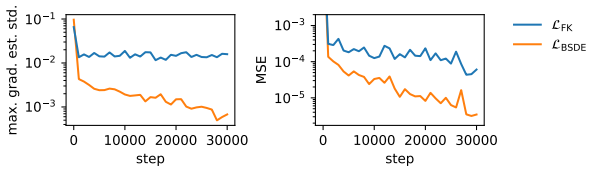


Figure 3. Gradient estimator standard deviations (maximum over directions $\phi = \partial_{\theta_i}\Phi_\theta$, where Φ_θ is a neural network with parameters θ, see Appendix E) and the MSE (on evaluation data) as a function of the gradient steps when solving the HJB equation from Section 3.3 with batch size $K = 128$. In line with Proposition 2.7 the standard deviation of the gradient of $\widehat{\mathcal{L}}_{\mathrm{BSDE}}^{(K)}$ decreases when the solution is sufficiently well approximated.

Note that even though the gradient estimator $\mathcal{L}_{\mathrm{FK}}^{(K)}$ does not have vanishing variance at the solution, a large batch size K can counteract this effect. We will study corresponding dependencies between variances and batch sizes in our numerical experiments in Section 3.

While Proposition 2.6 only makes a statement on the gradient variances at the solution $u = V$, the following proposition shall indicate that, at least close to the solution, one might still expect a small variance of the gradients of $\mathcal{L}_{\mathrm{BSDE}}^{(K)}$, see also Figure 3.

Proposition 2.7 (Stability of $\mathcal{L}_{\mathrm{BSDE}}$ close to the solution)**.** *Assume it holds almost surely that*

$$|u(\xi,\tau) - V(\xi,\tau)| \leq \varepsilon \quad \textit{and} \quad |\phi(\xi,\tau)| \leq \kappa \tag{22}$$

and that there exists $\gamma \in \mathbb{R}_{>0}$ *such that for every* $(x,t) \in \mathbb{R}^d \times [0,T]$ *it holds that*

$$\|\nabla_x u(x,t) - \nabla_x V(x,t)\| \leq \varepsilon(1 + \|x\|^\gamma) \tag{23}$$

and that

$$\|\nabla_x \phi(x,t)\| \leq \kappa(1 + \|x\|^\gamma). \tag{24}$$

Then the variance can be bounded by

$$\mathbb{V}\left[\frac{\delta}{\delta u}\mathcal{L}_{\mathrm{BSDE}}^{(K)}(u;\phi)\right] \leq \frac{C\varepsilon^2\kappa^2}{K}, \tag{25}$$

where $C \in \mathbb{R}_{>0}$ *is a constant that depends only on* γ, ξ, τ, *and the given PDE in* (1).

Proof. See Appendix B. □

Remark 2.8. We remark that the simultaneous approximation of a function and its gradient by a neural network, as required in (22) and (23), has, for instance, been considered in Berner et al. (2019). Using Proposition 2.2 we can see that it is actually sufficient to approximate the gradient $\nabla_x V$ (which only implies the approximation of V up to a constant) in order to reach zero variance. We also note that, for u far away from the solution V, the term S_u could potentially increase the estimator variance, see, e.g., Figure 14 in the appendix.

2.2. Algorithmic Details and Neural Network Approximations

So far our analysis has been agnostic of the choice of the approximating function space $\mathcal{U}$. In practice, we usually rely on functions represented by a neural network $\Phi_\theta \colon \mathbb{R}^{d+1} \to \mathbb{R}$ with parameters $\theta \in \mathbb{R}^p$. The training of such a network is then conducted by minimizing a suitable estimated and time-discretized loss $\widehat{\mathcal{L}}^{(K)}$ using some variant of gradient descent. In its simplest form, this corresponds to the update rule

$$\theta^{(m+1)} = \theta^{(m)} - \lambda \nabla_\theta \widehat{\mathcal{L}}^{(K)}(\Phi_{\theta^{(m)}}), \tag{26}$$

where $\lambda \in \mathbb{R}_{>0}$ is a learning rate and $\theta^{(0)} \in \mathbb{R}^p$ is a random initialization. Let us refer to Algorithm 1 in Appendix E for further details.

Relating to the Gateaux derivatives from the previous section, let us note that we are particularly interested in the directions[5] $\phi = \partial_{\theta_i}\Phi_\theta$ for $i \in \{1, \dots, p\}$. This choice is motivated by (26) and the chain rule of the Gateaux derivative, which, under suitable assumptions, states that

$$\partial_{\theta_i}\widehat{\mathcal{L}}^{(K)}(\Phi_\theta) = \frac{\delta}{\delta u}\Big|_{u=\Phi_\theta} \widehat{\mathcal{L}}^{(K)}\left(u; \partial_{\theta_i}\Phi_\theta\right). \tag{27}$$

Now, in order to elaborate on some computational aspects, let us state the actual gradients of the corresponding losses. For a neural network $u = \Phi_\theta$, we compute

$$\nabla_\theta \widehat{\mathcal{L}}^{(K)}_{\mathrm{FK}}(u) = -\frac{2}{K}\sum_{k=1}^{K} \widehat{\Delta}_u^{(k)} \nabla_\theta u, \tag{28a}$$

$$\nabla_\theta \widehat{\mathcal{L}}^{(K)}_{\mathrm{BSDE}}(u) = -\frac{2}{K}\sum_{k=1}^{K} e^{(k)} \left(\nabla_\theta u + \nabla_\theta \widehat{S}_u^{(k)}\right), \tag{28b}$$

where $e^{(k)} := \widehat{\Delta}_u^{(k)} - \widehat{S}_u^{(k)}$ is the sample-wise error.

From a computational point of view, note that the derivative of $\widehat{\mathcal{L}}^{(K)}_{\mathrm{FK}}$ as displayed in (28a) only relies on a single gradient computation of the approximating function u (w.r.t. the model parameters θ), whereas the derivative of $\widehat{\mathcal{L}}^{(K)}_{\mathrm{BSDE}}$ as displayed in (28b) needs evaluations of $\nabla_\theta s_j^{(k)}$, see (16) and (17). This might be costly for automatic differentiation (autodiff) tools and we will suggest computational speed-ups and memory-efficient versions in the next section.

Further, note that due to the term $\widehat{S}_u^{(k)}$ the loss $\widehat{\mathcal{L}}^{(K)}_{\mathrm{BSDE}}$ always needs to rely on discretized SDEs, whereas $\widehat{\mathcal{L}}^{(K)}_{\mathrm{FK}}$ can be computed without SDE simulations whenever explicit solutions of the stochastic processes are available (which, admittedly, is often not the case in practice – see, however, the examples in Sections 3.1 and 3.2).

[5] We assume that the functions Φ_θ, $\theta \in \mathbb{R}^p$, as well as all partial derivatives $\partial_{\theta_i}\Phi_\theta$ lie in $\mathcal{U}$.

2.3. Further Computational Adjustments

For the loss $\mathcal{L}_{\mathrm{BSDE}}$ as defined in (11) we can make further adjustments that will turn out to be relevant from a computational point of view. In particular, let us suggest two alternative versions that are based on the fact that the function u in (11) appears in two instances, which can be treated differently. One idea is to differentiate only with respect to one of its appearances. To this end, let us define the loss

$$\mathcal{L}^{\mathrm{detach}}_{\mathrm{BSDE}}(u, w) := \mathbb{E}\left[\left(\Delta_u - S_w\right)^2\right], \tag{29}$$

now depending on two arguments. We then compute derivatives according to

$$\frac{\delta}{\delta u}\Big|_{w=u} \mathcal{L}^{\mathrm{detach}}_{\mathrm{BSDE}}(u, w), \tag{30}$$

where in automatic differentiation tools setting $w = u$ after computing the derivative is usually obtained by detaching w from the computational graph[6]. One can readily check that $\mathcal{L}^{\mathrm{detach}}_{\mathrm{BSDE}}$ is a valid loss for problem (1) that keeps the robustness property stated in Propositions 2.4 and 2.6 intact. At the same time we expect a great reduction of the computational effort since the second-order derivatives of the form $\nabla_\theta \partial_{x_i} u$, appearing in $\nabla_\theta \widehat{S}_u^{(k)}$, do not need to be computed anymore – in particular note that multiple such derivatives would need to be evaluated in actual implementations due to the discretization of the integral, as stated in (16b). We refer to Section 3 where we can indeed demonstrate substantial computational improvements in several settings.

Another version of $\mathcal{L}_{\mathrm{BSDE}}$ relies on explicitly modelling the gradient of the approximating function u by an extra function, see also Zhou et al. (2021). We can therefore define the loss

$$\mathcal{L}^{\mathrm{grad}}_{\mathrm{BSDE}}(u, r) := \mathbb{E}\left[\left(\Delta_u - \widetilde{S}_r\right)^2\right], \tag{31}$$

where now

$$\widetilde{S}_r = \int_\tau^T \left(\sigma^\top r\right)(X_s, s) \cdot \mathrm{d}W_s. \tag{32}$$

The additional function $r : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ can be modelled with a separate neural network. The loss (31) is then minimized with respect to the parameters of the functions u and r simultaneously, which again avoids the computation of second-order derivatives.

Finally, let us introduce another viable option for decreasing computational resources, which is essentially a more memory-efficient way of computing the derivative formula stated in (28b). Usually, one performs a forward pass to

[6] In PyTorch and TensorFlow this can be achieved by the `detach` and `stop_gradient` operations.

compute $\widehat{\mathcal{L}}^{(K)}_{\mathrm{BSDE}}(u)$ and then uses automatic differentiation to obtain the gradient $\nabla_\theta \widehat{\mathcal{L}}^{(K)}_{\mathrm{BSDE}}(u)$. However, this requires to keep track of all the discretization steps for the stochastic integral $(s_j^{(k)})_{j=1}^{J^{(k)}}$, as they all depend on θ. This leads to a GPU memory consumption which scales linearly in the number of steps $J^{(k)}$ and, already for small K, exceeds common capacities, see Figures 5 and 7 in the appendix.

To circumvent this issue, we observe that the derivative in (28b) can be decomposed into

$$G_1 := -\frac{2}{K}\sum_{k=1}^{K} e^{(k)} \nabla_\theta u, \tag{33}$$

which equals the derivative of the loss $\mathcal{L}^{\mathrm{detach}}_{\mathrm{BSDE}}$, and

$$\sum_{j=1}^{J} G_{2,j} := \sum_{j=1}^{J} -\frac{2}{K}\sum_{k=1}^{K} e^{(k)} \nabla_\theta s_j^{(k)}, \tag{34}$$

where $J = \max_{k=1}^{K} J^{(k)}$ and $s_j^{(k)} = 0$ for $j > J^{(k)}$. Therefore, we can first compute and cache the errors $(e^{(k)})_{k=1}^{K}$ and the gradients G_1. Then, we repeat the same simulation, i.e., use the same realization of $(\xi^{(k)}, \tau^{(k)}, W^{(k)})$, to compute and accumulate the gradients $(G_{2,j})_{j=1}^{J}$ on-the-fly. In doing so, automatic differentiation only needs to track a single discretization step $s_j^{(k)}$. This keeps the memory footprint independent of the number of steps $J^{(k)}$ or, equivalently, the step-size $t_{j+1}^{(k)} - t_j^{(k)}$ – however, at the cost of an increased computational time[7]. Of course, the same trick can also be applied to the loss $\mathcal{L}^{\mathrm{grad}}_{\mathrm{BSDE}}$. Even though from a mathematical perspective the losses are still the same as $\mathcal{L}_{\mathrm{BSDE}}$ and $\mathcal{L}^{\mathrm{grad}}_{\mathrm{BSDE}}$, we give them the new names

$$\mathcal{L}_{\mathrm{BSDE,\,eff}} \qquad \text{and} \qquad \mathcal{L}^{\mathrm{grad}}_{\mathrm{BSDE,\,eff}} \tag{35}$$

in order to relate to their significant computational improvements, which we will illustrate in the next section.

3. Numerical Examples

In this section we aim to illustrate our theoretical results on three high-dimensional PDEs[8]. Specifically, we consider a heat equation with paraboloid terminal condition, a HJB equation arising in molecular dynamics, and a Black-Scholes model with correlated noise. The respective reference solutions are given as a closed-form expression, as a tensor-product of one-dimensional approximations obtained by finite-difference methods, and as a Monte-Carlo approximation using the Feynman-Kac formula, see Appendix C.

In all of our numerical experiments we use comparable setups which are detailed in Appendix E. We emphasize that, instead of using a train/val/test split on a given finite data set, our setting allows us to simulate new i.i.d. samples $(\xi^{(k)}, \tau^{(k)}, W^{(k)})$ for each batch on demand during training. To evaluate our methods, we compute the MSE between the neural network approximation and the reference solution at points (x, t) which are sampled independently of the training data, i.e., on previously unseen points, see Appendix E.

We will show that all considered losses are viable in the sense that they yield appropriate approximations of solutions to the associated PDEs. At the same time we will see substantial performance differences, which can be attributed to the different variance properties outlined before.

3.1. Heat Equation

Let us first consider a version of the heat equation with paraboloid terminal condition by setting $b(x,t) = 0$, $\sigma(x,t) = \bar{\sigma} \in \mathbb{R}^{d\times d}$, and $g(x) = \|x\|^2$. This allows for explicit solutions given by

$$V(x,t) = \|x\|^2 + \mathrm{Trace}(\bar{\sigma}\bar{\sigma}^\top)(T-t) \tag{36}$$

and

$$X_s = \xi + \bar{\sigma}(W_s - W_\tau), \tag{37}$$

see also Beck et al. (2021). We analyze the special case $\bar{\sigma} = \nu\ \mathrm{Id}$ with $\nu \in \mathbb{R}$, where we have that

$$\int_\tau^T \bar{\sigma}^\top \nabla_x V(X_s, s) \cdot \mathrm{d}W_s = 2\,\nu \int_\tau^T X_s \cdot \mathrm{d}W_s. \tag{38}$$

Using the fact that $\int_0^{\mathcal{T}} B_s \cdot \mathrm{d}B_s = \frac{1}{2}\big(\|B_{\mathcal{T}}\|^2 - d\mathcal{T}\big)$, where $\mathcal{T} = T - \tau$ and $B_s = W_{\tau+s} - W_\tau$, the stochastic integral in (38) evaluates to the following closed-form formula

$$2\nu\xi \cdot (W_T - W_\tau) + \nu^2\big(\|W_T - W_\tau\|^2 - d(T-\tau)\big). \tag{39}$$

This expression shows how the quantities in Propositions 2.4 and 2.6 may behave. For a fair comparison, however, we do not make use of this explicit representation.

For our experiments, we consider $d = 50$, $T = 1$, $\nu = 0.5$, $\xi \sim \mathrm{Unif}([-0.5, 0.5]^d)$, and $\tau \sim \mathrm{Unif}([0,1])$. For a fixed batch size all of our proposed methods significantly outperform the standard approach of minimizing $\mathcal{L}_{\mathrm{FK}}$, see Figure 1. For larger batch sizes the losses $\mathcal{L}_{\mathrm{BSDE}}$ and $\mathcal{L}^{\mathrm{grad}}_{\mathrm{BSDE}}$ run out of our memory limit of 8 GiB and the performance of $\mathcal{L}_{\mathrm{FK}}$ improves. However, when using the efficient versions or $\mathcal{L}^{\mathrm{detach}}_{\mathrm{BSDE}}$, we still outperform the baseline by a substantial margin. As suggested by our theory, the performance gains

[7] For most practical use-cases one is interested in obtaining the most accurate solution and there is only little restriction on the time for training, which needs to be done once per PDE. However, for time-critical applications, we present the performance w.r.t. the training time in Figures 11 and 12 in the appendix.

[8] The associated code can be found at https://github.com/juliusberner/robust_kolmogorov.

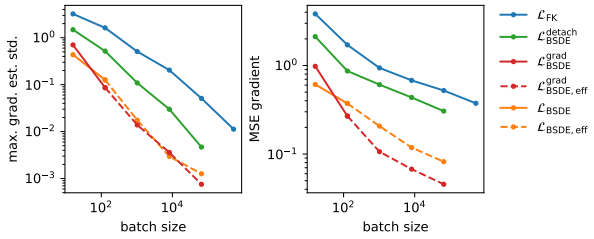


Figure 4. Standard deviation of the gradient estimators (maximum over directions $\phi = \partial_{\theta_i}\Phi_\theta$) and MSE of the gradients (on evaluation data) in the setup of Figure 1, see also Appendix E. In line with Proposition 2.7 and Remark 2.8, the standard deviations of the loss and gradient estimators are smaller for the BSDE-based losses, especially when the gradient $\nabla_x V$ is well approximated.

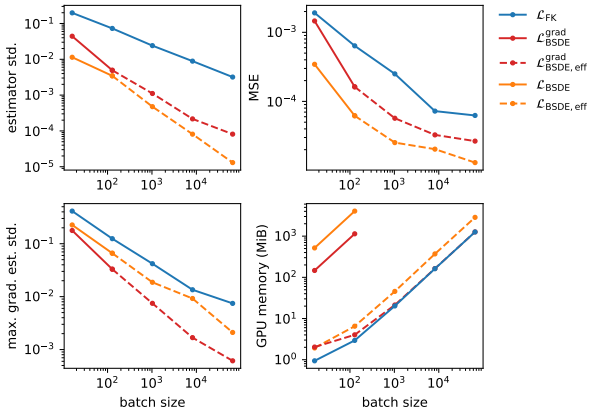


Figure 5. Estimator standard deviation, gradient estimator standard deviation (maximum over directions $\phi = \partial_{\theta_i}\Phi_\theta$), and MSE (on evaluation data) after $30k$ gradient steps for the Black–Scholes model in Section 3.2. The last plot shows that $\mathcal{L}_{\mathrm{BSDE}}$ and $\mathcal{L}^{\mathrm{grad}}_{\mathrm{BSDE}}$ cannot be used for batch size $K = 1024$ and larger as it would exceed the GPU memory limit of 8 GiB. The same holds for the other losses at batch size $K = 524288$.

are based on the variance reducing effect of the stochastic integral, see Figures 1 and 4. While the gradient $\nabla_x V$ can be best approximated by $\mathcal{L}^{\mathrm{grad}}_{\mathrm{BSDE,\,eff}}$ (see also Figure 8 in the appendix), which reduces the variance according to Proposition 2.7, the loss $\mathcal{L}_{\mathrm{BSDE,\,eff}}$ still performs better in approximating the solution V. This can be motivated by the fact that in case of $\mathcal{L}^{\mathrm{grad}}_{\mathrm{BSDE,\,eff}}$ one network learns the gradient and another one independently learns the solution. In contrast, in the case of $\mathcal{L}_{\mathrm{BSDE,\,eff}}$, a single network is trained to simultaneously approximate V and its gradient $\nabla_x V$. Note that these observations are consistent across various diffusivities ν, see Figures 9 and 10.

3.2. Black–Scholes Model with Correlated Noise

As a second example we consider a version of the celebrated Black–Scholes model from financial engineering (Black & Scholes, 1973), given by

$$\sigma(x,t) = \mathrm{diag}(\beta_1 x_1, \dots, \beta_d x_d)\bar{\sigma}, \quad b(x,t) = \bar{b}x, \quad (40)$$

with $\beta \in \mathbb{R}^d, \bar{\sigma} \in \mathbb{R}^{d\times d}, \bar{b} \in \mathbb{R}$, and

$$g(x) = \max\left\{0, \kappa - \min_{i=1}^{d} x_i\right\}. \quad (41)$$

It models a rainbow European put option, giving its holder the right to sell the minimum of underlying assets at the strike price $\kappa \in \mathbb{R}_{>0}$ at time T.

For the experiments we choose $d = 50$, $T = 1$, $\bar{b} = -0.05$, $\kappa = 5.5$, $\beta = (0.1 + i/(2d))_{i=1}^d$,

$$\xi \sim \mathrm{Unif}([4.5, 5.5]^d), \quad \tau \sim \mathrm{Unif}([0,1]), \quad (42)$$

and $\bar{\sigma}$ to be the lower triangular matrix arising from the Cholesky decomposition $\bar{\sigma}\bar{\sigma}^\top = Q$ with

$$Q_{i,j} = 0.5(1 + \delta_{i,j}), \quad i,j \in \{1,\dots,d\}, \quad (43)$$

where $\delta_{i,j}$ denotes the Kronecker delta. There does not exist a closed form solution V, however, one can use Monte Carlo sampling and the representation

$$(X_s)_i = \xi_i e^{\left(\bar{b} - \frac{\|\beta_i \sigma_i\|^2}{2}\right)(s-\tau) + \beta_i \sigma_i \cdot (W_s - W_\tau)}, \quad (44)$$

where σ_i is the i-th row of $\bar{\sigma}$, in order to evaluate the solution pointwise, see Beck et al. (2021). Note that, in case of the loss $\mathcal{L}_{\mathrm{FK}}$, such a closed-form expression could be used instead of the Euler-Maruyama scheme in order to speed up training. However, for most settings such a representation is not available and, for the sake of a fair comparison, we did not make use of it during training.

Note that for this example the coefficient functions σ and b depend on x, which makes the discretization of the SDE more delicate. The loss $\mathcal{L}^{\mathrm{detach}}_{\mathrm{BSDE}}$ can be sensitive to initial performance and performs suboptimally in this example, see Remark 2.8 and Figure 14. The other losses again underline our theoretical results and, in particular, substantially outperform the loss $\mathcal{L}_{\mathrm{FK}}$, see Figure 5. As suggested by our results, a larger batch size generally improves the robustness. However, the efficient versions of $\mathcal{L}_{\mathrm{BSDE}}$ and $\mathcal{L}^{\mathrm{grad}}_{\mathrm{BSDE}}$ do not need significantly more GPU memory than $\mathcal{L}_{\mathrm{FK}}$ such that they can be used with the same maximal batch size.

3.3. Hamilton–Jacobi–Bellman Equation

Finally, let us consider the nonlinear Hamilton–Jacobi–Bellman equation

$$(\partial_t + L)\widetilde{V}(x,t) + f(x,t) = \frac{1}{2}\left\|(\sigma^\top \nabla_x \widetilde{V})(x,t)\right\|^2 \quad (45)$$

for $(x,t) \in \mathbb{R}^d \times [0,T)$, with terminal condition

$$\widetilde{V}(x,T) = \widetilde{g}(x) \quad (46)$$

for $x \in \mathbb{R}^d$. This prominent PDE from control theory corresponds to the controlled stochastic process

$$\mathrm{d}X_s^v = (b(X_s^v, s) + (\sigma v)(X_s^v, s))\,\mathrm{d}s + \sigma(X_s^v, s)\,\mathrm{d}W_s,$$

where the control $v \in C(\mathbb{R}^d \times [0,T], \mathbb{R}^d)$ can be thought of as a steering force to be chosen so as to minimize a given cost function

$$J(v) = \mathbb{E}\left[\int_0^T \left(f_s + \frac{1}{2}\|v_s\|^2\right)\mathrm{d}s + \widetilde{g}(X_T^v)\right], \quad (47)$$

with the short-hands $f_s := f(X_s^v, s)$ and $v_s := v(X_s^v, s)$. In the above, $f \in C(\mathbb{R}^d \times [0,T], \mathbb{R})$ represents running costs (in addition to the squared costs on the control v) and $\widetilde{g} \in C(\mathbb{R}^d, \mathbb{R})$ specifies the terminal costs. It turns out that one can recover the optimal control v^* that minimizes the costs (47) from the solution to the PDE in (45) by the relation $v^* = -\sigma^\top \nabla_x \widetilde{V}$.

Now, with the transformation[9] $V = \exp(-\widetilde{V})$, we can convert the nonlinear PDE in (45) to a linear one of the form (1) (or, to be precise, of the form (69)) with boundary condition $g = \exp(-\widetilde{g})$, which allows us to apply Algorithm 1, see Lemma G.1 for the details.

For our experiments, we consider a problem that is prominent in molecular dynamics and has been suggested in Nüsken & Richter (2021b). We define the drift to be $b = -\nabla_x \Psi$, with $\Psi(x) = \kappa \sum_{i=1}^d (x_i^2 - 1)^2$ being a potential of double well type. In applications, the potential represents the energy associated to a system of atoms (i.e. molecule) and one is often interested in how molecular configurations change over time. This is then related to transition paths between metastable states of the stochastic dynamics, whose sampling poses formidable computational challenges, see e.g. Hartmann & Richter (2021).

For an example, let us define a terminal value that is supported in one of the 2^d minima of Ψ, namely $g(x) = \exp(-\eta \sum_{i=1}^d (x_i - 1)^2)$ and set $f = 0$. We choose $\eta = 0.04$ and $\kappa = 0.1$ in dimension $d = 10$ as well as $\sigma = \mathrm{Id}$. The left panel of Figure 6 shows the original potential Ψ as well as a tilted version that can be derived from the solution of the HJB equation (45) via $\Psi^* = \Psi + \sigma\sigma^\top \widetilde{V}$, computed with a finite difference reference method as well as with our algorithm. We can see that both solutions agree for all of our considered methods. However, note that for the optimal control functions the BSDE-based losses are again superior to $\mathcal{L}_{\mathrm{FK}}$, as displayed in the right panel. This is in line with the more accurate approximation of the gradient, which is demonstrated by a performance comparison in Figure 15 in the appendix.

[9] This transformation is also known as *Hopf-Cole transformation*, see, e.g., Section 4.4.1 in Evans (2010). See also Hartmann et al. (2017) for a discussion on related applications in importance sampling of stochastic processes relevant in computational statistical physics.

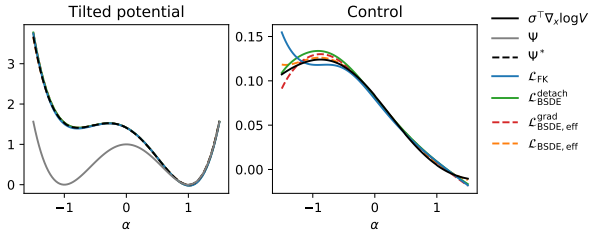


Figure 6. For a HJB equation with double well potential we display the tilted version of the potential Ψ^* evaluated at $x = (\alpha, \ldots, \alpha)^\top$ and $t = 0.75$, which is approximated well by all our methods (after $30k$ gradient steps with batch size 1024). However, the approximation of the first component of the control $\sigma^\top \nabla_x \log V$ depicted in the right panel is better for BSDE-based losses.

4. Conclusion

The most prominent method for solving general high-dimensional linear PDEs of Kolmogorov type is based on minimizing a variational formulation inspired by the Feynman-Kac formula over a class of neural networks. Building upon results on BSDEs and control variates for Monte Carlo estimators, we suggest an alternative formulation and prove that the corresponding loss estimator enjoys lower variance. Lending to the notion of Gateaux derivatives, we can further show that this leads to gradient estimators with lower variance, which is crucial for gradient-based optimization. Importantly, we also develop a novel, more efficient estimator that is adapted to current deep learning frameworks and show that resulting methods yield significant performance gains. It is up to future research to exploit such efficient estimators for boundary value problems, see Appendix C, and non-linear PDEs, which can naturally be approached with BSDE-based losses, see Appendix D.

In summary, we suspect that the techniques used in this paper can be transferred to a number of related deep learning methods that rely on Monte Carlo sampling. Providing important theoretical guarantees and performance improvements, our work is a fundamental step in developing robust and reliable algorithms for the solution of high-dimensional PDEs.

Acknowledgements

The research of Lorenz Richter has been funded by Deutsche Forschungsgemeinschaft (DFG) through the grant CRC 1114 “Scaling Cascades in Complex Systems” (project A05, project number 235221301). The research of Julius Berner was supported by the Austrian Science Fund (FWF) under grant I3403-N32. The computational results have been achieved in part using the Vienna Scientific Cluster (VSC).

References

Ash, R. B. and Doléans-Dade, C. *Probability and measure theory*. Academic press, 2000.

Baldi, P. *Stochastic Calculus: An Introduction Through Theory and Exercises*. Universitext. Springer International Publishing, 2017.

Beck, C., E, W., and Jentzen, A. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *Journal of Nonlinear Science*, 29(4):1563–1619, 2019.

Beck, C., Becker, S., Grohs, P., Jaafari, N., and Jentzen, A. Solving the Kolmogorov PDE by means of deep learning. *Journal of Scientific Computing*, 88(3):1–28, 2021.

Berner, J., Elbrächter, D., Grohs, P., and Jentzen, A. Towards a regularity theory for ReLU networks–chain rule and global error estimates. In *13th International conference on Sampling Theory and Applications*, pp. 1–5. IEEE, 2019.

Berner, J., Dablander, M., and Grohs, P. Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning. In *Advances in Neural Information Processing Systems*, pp. 16615–16627, 2020a.

Berner, J., Grohs, P., and Jentzen, A. Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black–Scholes partial differential equations. *SIAM Journal on Mathematics of Data Science*, 2(3):631–657, 2020b.

Berner, J., Grohs, P., Kutyniok, G., and Petersen, P. The modern mathematics of deep learning. *arXiv preprint arXiv:2105.04026*, 2021.

Bismut, J.-M. Conjugate convex functions in optimal stochastic control. *Journal of Mathematical Analysis and Applications*, 44(2):384–404, 1973.

Black, F. and Scholes, M. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3): 637–654, 1973.

Da Prato, G. and Zabczyk, J. *Stochastic equations in infinite dimensions*. Cambridge university press, 2014.

E, W., Han, J., and Jentzen, A. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

Ekström, E. and Tysk, J. The black–scholes equation in stochastic volatility models. *Journal of Mathematical Analysis and Applications*, 368(2):498 – 507, 2010.

Evans, L. C. *Partial differential equations*. American Mathematical Society, Providence, R.I., 2010.

Gall, J. *Brownian Motion, Martingales, and Stochastic Calculus*. Graduate Texts in Mathematics. Springer International Publishing, 2016.

Gobet, E. *Monte-Carlo methods and stochastic processes: from linear to non-linear*. CRC Press, 2016.

Grohs, P. and Herrmann, L. Deep neural network approximation for high-dimensional elliptic PDEs with boundary conditions. *arXiv preprint arXiv:2007.05384*, 2020.

Hairer, M., Hutzenthaler, M., and Jentzen, A. Loss of regularity for Kolmogorov equations. *The Annals of Probability*, 43(2):468–527, 2015.

Han, J., Jentzen, A., and Weinan, E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

Hartmann, C. and Richter, L. Nonasymptotic bounds for suboptimal importance sampling. *arXiv preprint arXiv:2102.09606*, 2021.

Hartmann, C., Richter, L., Schütte, C., and Zhang, W. Variational characterization of free energy: Theory and algorithms. *Entropy*, 19(11):626, 2017.

Hartmann, C., Kebiri, O., Neureither, L., and Richter, L. Variational approach to rare event simulation using least-squares regression. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(6):063107, 2019.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Huang, C.-W., Lim, J. H., and Courville, A. C. A variational perspective on diffusion-based generative models and score matching. *Advances in Neural Information Processing Systems*, 34, 2021.

Jentzen, A., Salimova, D., and Welti, T. A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients. *arXiv preprint arXiv:1809.07321*, 2018.

Karatzas, I. and Shreve, S. E. *Brownian Motion and Stochastic Calculus*. Springer, 1998.

Kiumarsi, B., Vamvoudakis, K. G., Modares, H., and Lewis, F. L. Optimal and autonomous control using reinforcement learning: A survey. *IEEE transactions on neural networks and learning systems*, 29(6):2042–2062, 2017.

Klenke, A. *Probability Theory: A Comprehensive Course*. Universitext. Springer London, 2013.

Kloeden, P. E. and Platen, E. Stochastic differential equations. In *Numerical Solution of Stochastic Differential Equations*, pp. 103–160. Springer, 1992.

Lelievre, T. and Stoltz, G. Partial differential equations and stochastic methods in molecular dynamics. *Acta Numerica*, 25:681, 2016.

Nüsken, N. and Richter, L. Interpolating between BSDEs and PINNs: deep learning for elliptic and parabolic boundary value problems. *arXiv preprint arXiv:2112.03749*, 2021a.

Nüsken, N. and Richter, L. Solving high-dimensional Hamilton–Jacobi–Bellman PDEs using neural networks: perspectives from the theory of controlled diffusions and measures on path space. *Partial Differential Equations and Applications*, 2(4):1–48, 2021b.

Øksendal, B. Stochastic differential equations. In *Stochastic differential equations*, pp. 65–84. Springer, 2003.

Pardoux, É. Backward stochastic differential equations and viscosity solutions of systems of semilinear parabolic and elliptic PDEs of second order. In *Stochastic Analysis and Related Topics VI*, pp. 79–127. Springer, 1998.

Pardoux, E. and Peng, S. Adapted solution of a backward stochastic differential equation. *Systems & Control Letters*, 14(1):55–61, 1990.

Pascucci, A. Kolmogorov equations in physics and in finance. In *Elliptic and parabolic problems*, pp. 353–364. Springer, 2005.

Pavliotis, G. A. *Stochastic processes and applications: diffusion processes, the Fokker-Planck and Langevin equations*, volume 60. Springer, 2014.

Pfau, D., Spencer, J. S., Matthews, A. G., and Foulkes, W. M. C. Ab initio solution of the many-electron Schrödinger equation with deep neural networks. *Physical Review Research*, 2(3):033429, 2020.

Pham, H. *Continuous-time stochastic control and optimization with financial applications*, volume 61. Springer Science & Business Media, 2009.

Pironneau, O. and Achdou, Y. Partial differential equations for option pricing. *Handbook of Numerical Analysis*, 15: 369–495, 2009.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Reisinger, C. and Zhang, Y. Rectified deep neural networks overcome the curse of dimensionality for nonsmooth value functions in zero-sum games of nonlinear stiff systems. *arXiv:1903.06652*, 2019.

Richter, L. *Solving high-dimensional PDEs, approximation of path space measures and importance sampling of diffusions*. PhD thesis, BTU Cottbus-Senftenberg, 2021.

Robert, C. P. and Casella, G. *Monte Carlo statistical methods*, volume 2. Springer, 2004.

Sirignano, J. and Spiliopoulos, K. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

Theodorou, E., Buchli, J., and Schaal, S. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research*, 11:3137–3181, 2010.

Vidales, M. S., Siska, D., and Szpruch, L. Unbiased deep solvers for parametric PDEs. *arXiv preprint arXiv:1810.05094*, 2018.

Widder, D. V. *The heat equation*, volume 67. Academic Press, 1976.

Zhang, J. *Backward stochastic differential equations*. Springer, 2017.

Zhou, M., Han, J., and Lu, J. Actor-critic method for high dimensional static Hamilton–Jacobi–Bellman partial differential equations based on neural networks. *arXiv preprint arXiv:2102.11379*, 2021.

A. Notation

We write $C^{2,1}(\mathbb{R}^d \times [0,T], \mathbb{R})$ for the space of functions $(x,t) \mapsto f(x,t)$ which are twice continuously differentiable in the spatial coordinate $x \in \mathbb{R}^d$ and once continuously differentiable in the time coordinate $t \in [0,T]$. We further denote the partial derivatives (w.r.t. to the i-th spatial coordinate x_i and time coordinate t) and the gradient of f (w.r.t. to the spatial coordinate x) by $\partial_{x_i} f$, $\partial_t f$, and $\nabla_x f = (\partial_{x_1} f, \dots, \partial_{x_d} f)^\top$, respectively. We say that functions $f \in C(\mathbb{R}^d \times [0,T], \mathbb{R})$ and $g \in C(\mathbb{R}^d, \mathbb{R})$ are *at most polynomially growing* if there exist constants $c, \lambda \in \mathbb{R}_{>0}$ such that for every $(x,t) \in \mathbb{R}^d \times [0,T]$ it holds that

$$|g(x)| \leq c(1 + \|x\|^\lambda), \quad |f(x,t)| \leq c(1 + \|x\|^\lambda). \tag{48}$$

Finally, we write $\|\cdot\|$ for the Euclidean norm and $\|\cdot\|_F$ for the Frobenius norm.

B. Assumptions and Proofs

We usually operate under the following assumption which guarantees a unique strong solution to the SDE in (6) as well as the existence of a unique strong solution $V \in C^{2,1}(\mathbb{R}^d \times [0,T], \mathbb{R})$ to the PDE in (1) which can be expressed as $V(x,t) = \mathbb{E}[g(X_T)|X_t = x]$ as, for instance, stated in (13), see Baldi (2017, Theorem 10.6), Karatzas & Shreve (1998, Remark 5.7.8), Øksendal (2003, Theorem 5.2.1), and Pavliotis (2014, Theorem 2.1).

Assumption 1 (Conditions on b, σ, and g)**.** *We assume that $g \in C^2(\mathbb{R}^d, \mathbb{R})$ is an at most polynomially growing function. Further, we assume that the coefficient functions $\sigma \in C(\mathbb{R}^d \times [0,T], \mathbb{R}^{d\times d})$ and $b \in C(\mathbb{R}^d \times [0,T], \mathbb{R}^d)$ satisfy the following properties: There exist constants $c_1, c_2, c_3, c_4 \in \mathbb{R}_{>0}$ such that*

$$\|b(x,t)\| + \|\sigma\sigma^\top(x,t)\|_F \leq c_1, \quad \textit{(boundedness)}$$
$$\|\sigma(x,t) - \sigma(y,t)\|_F \leq c_2\|x-y\|, \quad \textit{(Lipschitz continuity)}$$
$$\|b(x,t) - b(y,t)\| \leq c_3\|x-y\|, \quad \textit{(Lipschitz continuity)}$$
$$\eta \cdot (\sigma\sigma^\top)(x,t)\eta \geq c_4\|\eta\|^2, \quad \textit{(uniform ellipticity)}$$

for all $x, y, \eta \in \mathbb{R}^d$ and $t \in [0,T]$.

Throughout the proofs, we denote by $\Delta_u^{(k)}$ and $S_u^{(k)}$ the quantities in (8a) and (8b) evaluated at the k-th sample. This corresponds to replacing (ξ, τ, W) in (6), (8a), and (8b) by the k-th sample $(\xi^{(k)}, \tau^{(k)}, W^{(k)})$.

Proof of Proposition 2.2. We will use the fact that $\Delta_V = S_V$ almost surely, see Lemma 2.1. This allows us to rewrite

$$\begin{aligned}\mathcal{L}_{\mathrm{BSDE}}(u) &= \mathbb{E}\left[(\Delta_u + V(\xi,\tau) - V(\xi,\tau) - S_u)^2\right] \\ &= \mathbb{E}\left[(V(\xi,\tau) - u(\xi,\tau) + S_V - S_u)^2\right] \\ &= \mathbb{E}\left[(V(\xi,\tau) - u(\xi,\tau))^2\right] + \mathbb{E}\left[S_{V-u}^2\right].\end{aligned}$$

In the last step we used the tower property of the conditional expectation and the vanishing expectation of the stochastic integrals S_V and S_u to show that the cross-term has zero expectation, namely

$$\begin{aligned}&\mathbb{E}\left[(V(\xi,\tau) - u(\xi,\tau))(S_V - S_u)\right] \\ &\quad = \mathbb{E}\left[(V(\xi,\tau) - u(\xi,\tau))\,\mathbb{E}\left[S_V - S_u|(\xi,\tau)\right]\right] = 0.\end{aligned}$$

For the loss $\mathcal{L}_{\mathrm{FK}}$ we can similarly write

$$\begin{aligned}\mathcal{L}_{\mathrm{FK}}(u) &= \mathbb{E}\left[(V(\xi,\tau) - u(\xi,\tau) + \Delta_V)^2\right] &\text{(49a)}\\ &= \mathbb{E}\left[(V(\xi,\tau) - u(\xi,\tau))^2\right] + \mathbb{E}\left[\Delta_V^2\right], &\text{(49b)}\end{aligned}$$

where the cross-term has again zero expectation. More precisely, the tower property of the conditional expectation and Lemma 2.1 imply that

$$\begin{aligned}&\mathbb{E}\left[(V(\xi,\tau) - u(\xi,\tau))\,\Delta_V\right] &\text{(50a)}\\ &\quad = \mathbb{E}\left[(V(\xi,\tau) - u(\xi,\tau))\,\mathbb{E}\left[\Delta_V|(\xi,\tau)\right]\right] &\text{(50b)}\\ &\quad = \mathbb{E}\left[(V(\xi,\tau) - u(\xi,\tau))\,\mathbb{E}\left[S_V|(\xi,\tau)\right]\right] = 0. &\text{(50c)}\end{aligned}$$

This proves Proposition 2.2. □

Proof of Proposition 2.4. Let us recall Lemma 2.1 which states that $\Delta_V = S_V$ almost surely. With the mutual independence of $(\Delta_u^{(k)})_{k=1}^K$ and $(S_u^{(k)})_{k=1}^K$ we now readily obtain that

$$\mathbb{E}\left[\mathcal{L}_{\mathrm{BSDE}}^{(K)}(V)\right] = \mathbb{E}\left[(\Delta_V - S_V)^2\right] = 0, \tag{51}$$

and

$$\mathbb{V}\left[\mathcal{L}_{\mathrm{BSDE}}^{(K)}(V)\right] = \frac{1}{K}\,\mathbb{V}\left[(\Delta_V - S_V)^2\right] = 0. \tag{52}$$

For the loss $\mathcal{L}_{\mathrm{FK}}$, it holds that

$$\mathbb{E}\left[\mathcal{L}_{\mathrm{FK}}^{(K)}(V)\right] = \mathbb{E}\left[\Delta_V^2\right] = \mathbb{V}\left[S_V\right] \tag{53}$$

and

$$\mathbb{V}\left[\mathcal{L}_{\mathrm{FK}}^{(K)}(V)\right] = \frac{1}{K}\,\mathbb{V}\left[\Delta_V^2\right] = \frac{1}{K}\,\mathbb{V}\left[S_V^2\right], \tag{54}$$

which proves the claim. □

Proof of Proposition 2.6. (i) It holds that

$$\begin{aligned}\frac{\delta}{\delta u}\mathcal{L}_{\mathrm{FK}}^{(K)}(u;\phi) &= \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0}\mathcal{L}_{\mathrm{FK}}^{(K)}(u+\varepsilon\phi) &\text{(55a)}\\ &= \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0}\frac{1}{K}\sum_{k=1}^K\left(\Delta_{u+\varepsilon\phi}^{(k)}\right)^2 &\text{(55b)}\\ &= -\frac{2}{K}\sum_{k=1}^K \Delta_u^{(k)}\phi(\xi^{(k)},\tau^{(k)}). &\text{(55c)}\end{aligned}$$

Using the mutual independence of our samples this implies that

$$\mathbb{V}\left[\frac{\delta}{\delta u}\mathcal{L}_{\mathrm{FK}}^{(K)}(u;\phi)\right] = \frac{4}{K}\,\mathbb{V}\left[\Delta_u\phi(\xi,\tau)\right]. \tag{56}$$

Now, setting $u = V$, Lemma 2.1 shows that the variance of the Gateaux derivative at V satisfies

$$\mathbb{V}\left[\frac{\delta}{\delta u}\Big|_{u=V}\mathcal{L}_{\mathrm{FK}}^{(K)}(u;\phi)\right] = \frac{4}{K}\,\mathbb{V}\left[S_V\phi(\xi,\tau)\right]. \tag{57}$$

By the Itô isometry, this evaluates to

$$\frac{4}{K}\,\mathbb{E}\left[\left(\int_\tau^T \|\sigma^\top\nabla_x V(X_s,s)\|^2\mathrm{d}s\right)\phi^2(\xi,\tau)\right], \tag{58}$$

which, in general, is non-zero for an arbitrary $\phi \in \mathcal{U}$.

(ii) By definition, we have that

$$\mathcal{L}_{\mathrm{BSDE}}^{(K)}(u+\varepsilon\phi) = \frac{1}{K}\sum_{k=1}^K\left(\Delta_{u+\varepsilon\phi}^{(k)} - S_{u+\varepsilon\phi}^{(k)}\right)^2. \tag{59}$$

The Gateaux derivative

$$\frac{\delta}{\delta u}\mathcal{L}_{\mathrm{BSDE}}^{(K)}(u;\phi) = \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\Big|_{\varepsilon=0}\mathcal{L}_{\mathrm{BSDE}}^{(K)}(u+\varepsilon\phi) \tag{60}$$

thus evaluates to

$$-\frac{2}{K}\sum_{k=1}^K\left(\Delta_u^{(k)} - S_u^{(k)}\right)\left(\phi(\xi^{(k)},\tau^{(k)}) + S_\phi^{(k)}\right). \tag{61}$$

Now, setting $u = V$, Lemma 2.1 readily implies that

$$\frac{\delta}{\delta u}\Big|_{u=V}\mathcal{L}_{\mathrm{BSDE}}^{(K)}(u;\phi) = 0 \tag{62}$$

almost surely for all $\phi \in \mathcal{U}$. This shows that

$$\mathbb{V}\left[\frac{\delta}{\delta u}\Big|_{u=V}\mathcal{L}_{\mathrm{BSDE}}^{(K)}(u;\phi)\right] = 0, \tag{63}$$

which proves the claim. Note that the above calculation shows that the derivative of any squared loss exhibits zero variance whenever the error is vanishing almost surely. □

Proof of Proposition 2.7. Let us define

$$\delta(x,t) = u(x,t) - V(x,t). \tag{64}$$

Analogously to (61) we can compute

$$\begin{aligned}\mathbb{V}\left[\frac{\delta}{\delta u}\Big|_{u=V+\delta}\mathcal{L}_{\mathrm{BSDE}}^{(K)}(u;\phi)\right]\\ = \frac{4}{K}\,\mathbb{V}\left[\left(\delta(\xi,\tau)+S_\delta\right)\left(\phi(\xi,\tau)+S_\phi\right)\right].\end{aligned} \tag{65}$$

Now, properties of the variance as well as the Cauchy-Schwarz inequality yield

$$\begin{aligned}&\mathbb{V}\left[\left(\delta(\xi,\tau)+S_\delta\right)\left(\phi(\xi,\tau)+S_\phi\right)\right]\\ &\quad\le \mathbb{E}\left[\left(\delta(\xi,\tau)+S_\delta\right)^2\left(\phi(\xi,\tau)+S_\phi\right)^2\right]\\ &\quad\le \mathbb{E}\left[\left(\delta(\xi,\tau)+S_\delta\right)^4\right]^{\frac{1}{2}}\mathbb{E}\left[\left(\phi(\xi,\tau)+S_\phi\right)^4\right]^{\frac{1}{2}}.\end{aligned}$$

Each of the above factors can be bounded by using the Burkholder-Davis-Gundy inequality (Da Prato & Zabczyk, 2014, Section 4.6), Hölder's inequality, and the consistency of the Euclidean norm, i.e.

$$\begin{aligned}&\mathbb{E}\left[\left(\delta(\xi,\tau)+S_\delta\right)^4\right] \le 8\left(\mathbb{E}\left[\delta(\xi,\tau)^4\right] + \mathbb{E}\left[S_\delta^4\right]\right)\\ &\le 8\Big(\varepsilon^4 + 36T\int_0^T\mathbb{E}\left[\|(\sigma^\top\nabla_x\delta)(X_s,s)\|^4\right]\mathrm{d}s\Big)\\ &\le 8\varepsilon^4\Big(1+36T\int_0^T\mathbb{E}\left[\|\sigma(X_s,s)\|_F^4(1+\|X_s\|^\gamma)^4\right]\mathrm{d}s\Big).\end{aligned}$$

Defining

$$C \coloneqq 32\Big(1+36T\int_0^T\mathbb{E}\left[\|\sigma(X_s,s)\|_F^4(1+\|X_s\|^\gamma)^4\right]\mathrm{d}s\Big),$$

we thus showed that

$$\mathbb{V}\left[\frac{\delta}{\delta u}\Big|_{u=V+\delta}\mathcal{L}_{\mathrm{BSDE}}^{(K)}(u;\phi)\right] \le \frac{C\varepsilon^2\kappa^2}{K}, \tag{66}$$

which proves the claim. □

C. Feynman-Kac Theorem and More General Linear PDEs

As outlined in Section 2 and for instance stated in (5) as well as (13), the Feynman-Kac formula brings a stochastic representation of the linear PDE in (1) via

$$\begin{aligned}V(x,t) &= \mathbb{E}[g(X_T)|X_t = x] &\text{(67a)}\\ &= \mathbb{E}[g(X_T)|(\xi,\tau) = (x,t)]. &\text{(67b)}\end{aligned}$$

Note that this can be shown using the identity in (9). More specifically, the stochastic integral S_V has a vanishing expectation conditioned on (ξ,τ) and we obtain that

$$\mathbb{E}[\Delta_V - S_V|(\xi,\tau)] = \mathbb{E}[\Delta_V|(\xi,\tau)] = 0. \tag{68}$$

Let us make this observation precise and at the same time state a slightly more general version of the Feynman-Kac theorem.

Theorem C.1 (Feynman-Kac formula)**.** *Let $g \in C^2(\mathbb{R}^d,\mathbb{R})$, $k \in C(\mathbb{R}^d\times[0,T],\mathbb{R})$, and $V \in C^{2,1}(\mathbb{R}^d\times[0,T],\mathbb{R})$ be at most polynomially growing functions. Further, let*

$f \in C(\mathbb{R}^d \times [0,T], \mathbb{R}_{\geq 0})$ and assume that V solves the parabolic problem

$$(\partial_t + L - f(x,t))V(x,t) + k(x,t) = 0 \tag{69}$$

on $(x,t) \in \mathbb{R}^d \times [0,T)$ with terminal condition

$$V(x,T) = g(x), \qquad x \in \mathbb{R}^d. \tag{70}$$

Then

$$V(x,t) = \mathbb{E}\left[\int_t^T e^{-\int_t^r f(X_s,s)\mathrm{d}s} k(X_r,r)\mathrm{d}r + e^{-\int_t^T f(X_s,s)\mathrm{d}s} g(X_T)\middle| X_t = x\right], \tag{71}$$

where X is a strong solution to (6).

Proof. The proof, whose main ingredient is Itô's Lemma, can, for instance, be found in Karatzas & Shreve (1998, Theorem 5.7.6) and Baldi (2017, Theorem 10.5). □

Let us recall that our loss $\mathcal{L}_{\mathrm{FK}}$ as defined in (10) readily follows from the Feynman-Kac formula – see also Beck et al. (2021). Note that a crucial point in the derivation of the Feynman-Kac formula is the elimination of the stochastic integral S_V, as defined in (8b), via its martingale property, see (68). Ironically, as elaborated on in Section 2, it turns out that this elimination is just the reason for larger variances of estimators of $\mathcal{L}_{\mathrm{FK}}$.

Remark C.2 (Initial value problems). Note that time can be reversed and initial value problems (as opposed to terminal value problems as in Theorem C.1) can be formulated. As an example, we can consider the time-homogeneous case, where b, σ, f, and k do not depend on time. Let $V \in C^{2,1}(\mathbb{R}^d \times [0,T], \mathbb{R})$ solve the parabolic PDE

$$(\partial_t - L + f(x))V(x,t) - k(x) = 0 \tag{72}$$

on $(x,t) \in \mathbb{R}^d \times (0,T]$ with initial condition

$$V(x,0) = g(x), \qquad x \in \mathbb{R}^d. \tag{73}$$

The associated stochastic representation is then given by

$$V(x,t) = \mathbb{E}\left[\int_0^t e^{-\int_0^r f(X_s)\mathrm{d}s} k(X_r)\mathrm{d}r + e^{-\int_0^t f(X_s)\mathrm{d}s} g(X_t)\middle| X_0 = x\right]. \tag{74}$$

Remark C.3 (Bounded domains). We can restrict ourselves to open, bounded domains $\mathcal{D} \subset \mathbb{R}^d$ and consider the parabolic PDE in (69) on $\mathcal{D}$, adding the additional boundary condition[10] $V(x,t) = g(x)$ for $(x,t) \in \partial\mathcal{D} \times [0,T]$. The stochastic representation then becomes

$$V(x,t) = \mathbb{E}\left[\int_t^{\mathcal{T}\wedge T} e^{-\int_t^r f(X_s,s)\mathrm{d}s} k(X_r,r)\mathrm{d}r + e^{-\int_t^{\mathcal{T}\wedge T} f(X_s,s)\mathrm{d}s} g(X_{\mathcal{T}\wedge T})\middle| X_t = x\right],$$

where $T \wedge \mathcal{T} := \min\{T, \mathcal{T}\}$ and $\mathcal{T} := \inf\{t \geq 0 : X_t \notin \mathcal{D}\}$ is the first exit time of the stochastic process from the domain $\mathcal{D}$, for which we usually assume $\mathcal{T} < \infty$ almost surely. Likewise, we can consider the elliptic boundary value problem

$$(L - f(x))V(x) + k(x) = 0, \qquad x \in \mathcal{D}, \tag{75a}$$

$$V(x) = g(x), \qquad x \in \partial\mathcal{D}, \tag{75b}$$

where now the solution V, the coefficients b and σ in the SDE (6), as well as f and k do not depend explicitly on time anymore, yielding the stochastic representation

$$V(x) = \mathbb{E}\left[\int_0^{\mathcal{T}} e^{-\int_0^r f(X_s)\mathrm{d}s} k(X_r)\mathrm{d}r + e^{-\int_0^{\mathcal{T}} f(X_s)\mathrm{d}s} g(X_{\mathcal{T}})\middle| X_0 = x\right], \tag{76}$$

again with $\mathcal{T} := \inf\{t \geq 0 : X_t \notin \mathcal{D}\}$, see e.g. Proposition 5.7.2 in Karatzas & Shreve (1998).

Leveraging the above representations, our proposed methods can readily be applied to a range of elliptic and parabolic PDEs on bounded domains. Note, however, that one needs to take into account the hitting times $\mathcal{T}$, for instance, using naive stopping criteria (Nüsken & Richter, 2021a) or more elaborate walk-on-the-sphere algorithms (Grohs & Herrmann, 2020). As this might obscure the comparisons between different loss functions, we focus on unbounded domains in our experiments.

D. Backward Stochastic Differential Equations and Semi-Linear PDEs

Backward stochastic differential equations (BSDEs) have been studied extensively in the last three decades and we refer to Pardoux (1998), Pham (2009), Gobet (2016), and Zhang (2017) for good introductions to the topic. Even though in this paper we only consider linear PDEs as stated in (1) or (69), BSDEs are typically associated to nonlinear (parabolic) PDEs of the type

$$(\partial_t + L)V(x,t) = h(x,t,V(x,t),(\sigma^\top \nabla_x V)(x,t)) \tag{77}$$

[10] One can also consider boundary conditions that are different from the terminal condition in (70), see, for instance, Baldi (2017, Theorem 10.4) and Lelievre & Stoltz (2016, Proposition 6.1).

for $(x,t) \in \mathbb{R}^d \times [0,T]$, a nonlinearity $h : \mathbb{R}^d \times [0,T] \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$, and differential operator L defined as in (2). The terminal value is given by

$$V(x,T) = g(x), \tag{78}$$

for a specified function $g \in C(\mathbb{R}^d, \mathbb{R})$.

BSDEs were first introduced in Bismut (1973) and their systematic study began with Pardoux & Peng (1990). Loosely speaking, they can be understood as nonlinear extensions of the Feynman-Kac formula (see Pardoux (1998) and Appendix C), relating the nonlinear PDE in (77) to the stochastic process X_s defined by

$$\mathrm{d}X_s = b(X_s, s)\,\mathrm{d}s + \sigma(X_s, s)\,\mathrm{d}W_s, \quad X_0 = x_0. \tag{79}$$

The key idea is then to define the processes

$$Y_s = V(X_s, s), \qquad Z_s = (\sigma^\top \nabla_x V)(X_s, s), \tag{80}$$

as representations of the PDE solution and its gradient, respectively, and apply Itô's lemma to obtain

$$\mathrm{d}Y_s = h(X_s, s, Y_s, Z_s)\,\mathrm{d}s + Z_s \cdot \mathrm{d}W_s, \tag{81}$$

with terminal condition $Y_T = g(X_T)$. Noting that the processes Y and Z are adapted[11] to the filtration generated by the Brownian motion W, they should indeed be understood as backward processes and not be confused with time-reversed processes. A convenient interpretation of the relations in (80) is that solving for the processes Y and Z under the constraint (81) corresponds to determining the solution of the PDE in (77) (and its gradient) along a random grid which is provided by the stochastic process X defined in (79).

Let us note that under suitable assumptions on the coefficients b, σ, h, and g one can prove existence and uniqueness of a solution to the BSDE system as defined in (79) and (81), see for instance Theorem 4.3.1 in Zhang (2017).

We further note that the standard BSDE system can be generalized to

$$\begin{aligned}\mathrm{d}X_s^v &= (b(X_s^v, s) + v(X_s^v, s))\,\mathrm{d}s + \sigma(X_s^v, s)\mathrm{d}W_s,\\ \mathrm{d}Y_s^v &= (h(X_s^v, s, Y_s^v, Z_s^v) + v(X_s^v, s)\cdot Z_s^v)\mathrm{d}s + Z_s^v \cdot \mathrm{d}W_s,\end{aligned}$$

with

$$X_0^v = x, \qquad Y_T^v = g(X_T^v). \tag{82}$$

In the above, $v \in C(\mathbb{R}^d \times [0,T], \mathbb{R}^d)$ is a suitable control vector field that can be understood as pushing the forward trajectories into desired regions of the state space, noting that the relations

$$Y_s^v = V(X_s^v, s), \qquad Z_s^v = (\sigma^\top \nabla_x V)(X_s^v, s), \tag{83}$$

[11] Intuitively, this means that the processes Y and Z must not depend on future values of the Brownian motion W.

Algorithm 1 Solving the PDE in (1) via deep learning

input Neural network Φ with initial parameters $\theta^{(0)}$, optimizer method step for updating the parameters, maximum number of steps M, batch size K, step-size Δt
output parameters $\theta^{(M)}$
for $m \leftarrow 0, \dots, M-1$ **do**
  $(\xi^{(k)}, \tau^{(k)}, W^{(k)})_{k=1}^K \leftarrow$ sample from $(\xi, \tau, W)^{\otimes K}$
  $(\widehat{X}^{(k)})_{k=1}^K \leftarrow$ simulate using the EM scheme in (84)
  $\mathcal{L} \leftarrow$ pick $\mathcal{L} \in \{\mathcal{L}_{\mathrm{FK}}, \mathcal{L}_{\mathrm{BSDE}}\}$
  $\widehat{\mathcal{L}}^{(K)}(\Phi_{\theta^{(m)}}) \leftarrow$ compute estimator loss as in (15)
  $\nabla_\theta \widehat{\mathcal{L}}^{(K)}(\Phi_{\theta^{(m)}}) \leftarrow \mathrm{autodiff}(\widehat{\mathcal{L}}^{(K)}(\Phi_{\theta^{(m)}}))$
  $\theta^{(m+1)} \leftarrow \mathrm{step}\left(\theta^{(m)}, \nabla_\theta \widehat{\mathcal{L}}^{(K)}(\Phi_{\theta^{(m)}})\right)$
end for

with $V \in C^{2,1}(\mathbb{R}^d \times [0,T], \mathbb{R})$ being the solution to the parabolic PDE in (77), hold true independent of the choice of v (Hartmann et al., 2019).

E. Further Computational Details

For convenience, we first summarize our considered method in Algorithm 1. In numerical simulations we need to discretize the stochastic process X as defined in (6) on a time grid $\tau = t_1 < \cdots < t_J$. A practical way to do so is based on the Euler-Maruyama (EM) scheme

$$\widehat{X}_{j+1} = \widehat{X}_j + b(\widehat{X}_j, t_j)\Delta t + \sigma(\widehat{X}, t_j)\sqrt{\Delta t}\,\zeta_{j+1}, \tag{84}$$

where $\Delta t \coloneqq t_{j+1} - t_j$ is the step-size and

$$\zeta_{j+1} \coloneqq \frac{W_{t_{j+1}} - W_{t_j}}{\sqrt{\Delta t}} \sim \mathcal{N}(0, \mathrm{Id}) \tag{85}$$

is a standard normally distributed random variable. It can be shown that $\widehat{X}_j$ convergences to $X_{j\Delta t}$ in an appropriate sense (Kloeden & Platen, 1992). This readily leads to discrete versions of the quantities Δ_u and S_u as defined in (16). Note that the discrete process $\widehat{X}$ is initialized at the random value $\widehat{X}_1 = \xi$ and J is chosen according to the randomly drawn initial time τ. More precisely, we set $J = \lceil (T-\tau)/\Delta t \rceil$ and use a smaller final step-size in order to arrive at the terminal time T. An alternative strategy would be to fix J for all realizations and change the step-size $\Delta t = \frac{T-\tau}{J}$ depending on the value of τ. We display the memory requirements of our methods as a function of the step-size Δt in Figure 7.

In Table 1 we summarize the hyperparameters for our experiments. We specify the amount of steps (Schedule A) or time (Schedule B) that we train and the percentile, i.e., milestone, where we decrease the learning rate λ and the step-size Δt. In order to have comparable results, we set a GPU memory limit of 8 GiB during training and always use batch sizes $K \in \{2^4, 2^7, 2^{10}, 2^{13}, 2^{16}, 2^{19}\}$.

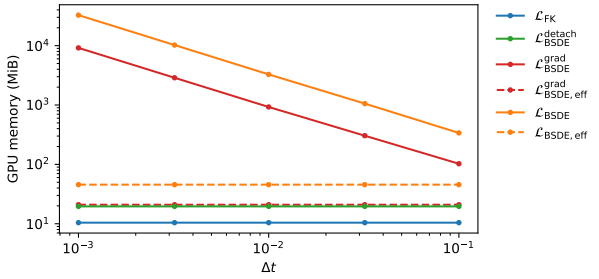


Figure 7. GPU memory requirements for a gradient step with batch size $K = 1024$ and different step-sizes Δt for the SDE discretization when solving the heat equation. If we do not consider the efficient versions, the memory usage for the losses $\mathcal{L}^{\mathrm{grad}}_{\mathrm{BSDE}}$ and $\mathcal{L}_{\mathrm{BSDE}}$ depends approximately linearly on Δt.

We employ a Multilevel neural network architecture, which has shown to be advantageous over standard feed-forward architectures. More precisely, we use an architecture with $L = 3$ levels, amplifying factor $q = 5$ for the HJB equation and $q = 3$ for the other PDEs, intermediate residual connections ($\chi = 1$) and without normalization layer, as defined in Berner et al. (2020a). Note that in case of the losses $\mathcal{L}^{\mathrm{grad}}_{\mathrm{BSDE}}$ and $\mathcal{L}^{\mathrm{grad}}_{\mathrm{BSDE,\,eff}}$ we use the same architecture with output dimension d for the neural network representing the function r in (31). As the activation function needs to be twice differentiable for certain losses, we replace the ReLU by the SiLU (also known as swish) activation function (Hendrycks & Gimpel, 2016).

Finally, Table 1 also specifies the number of samples used to approximate the MSE metrics and the number of batches used to estimate the loss and gradient variances. For the latter, we compute the value and derivative of $\widehat{\mathcal{L}}^{(K)}$ (w.r.t. the parameters θ of Φ_θ) for 30 batches consisting of K independent samples of (ξ, τ, W) by performing forward and backward passes without updating the parameters θ. To compute the MSE we evaluate $\widehat{\mathcal{L}}^{(N)}_{\mathrm{Eval}}$ with $N = 10 \cdot 2^{17}$ i.i.d. samples drawn from the distribution of (ξ, τ) – independently of the training data and independently for each evaluation. In this sense, we always evaluate our model w.r.t. to the real solution V on unseen data. For comparing the gradients we replace $(u(\xi,\tau) - V(\xi,\tau))^2$ by $\|\nabla_x u(\xi,\tau) - \nabla_x V(\xi,\tau)\|^2$ or $\|r(\xi,\tau) - \nabla_x V(\xi,\tau)\|^2$ in case of the methods $\mathcal{L}^{\mathrm{grad}}_{\mathrm{BSDE}}$ and $\mathcal{L}^{\mathrm{grad}}_{\mathrm{BSDE,\,eff}}$. When no closed-form solution for V is available, as for the Black–Scholes model in Section 3.2, we use the version in (7b) with another 2^{16} independent samples to estimate the inner expectation.

F. Further Numerical Experiments

Figures 8 to 15 show the results of additional numerical experiments. The corresponding captions describe the respective settings.

Table 1. Training and evaluation setup

Schedule A: Step limit	
steps M	$3 \cdot 10^4$
milestone	0.9
Schedule B: Time limit	
time	$24h$
milestone	0.5
Training	
learning rate λ	$[5 \cdot 10^{-4}, 5 \cdot 10^{-6}]$
step-size Δt	$[10^{-2}, 10^{-3}]$
optimizer	Adam
batch size K	$\{2^4, 2^7, 2^{10}, 2^{13}, 2^{16}, 2^{19}\}$
GPU memory constraint	8 GiB
Network	
architecture	Multilevel
(L, q, χ)	$\{(3,3,1), (3,5,1)\}$
activation function	SiLU
Evaluation	
samples N	$10 \cdot 2^{17}$
batches (for variances)	30

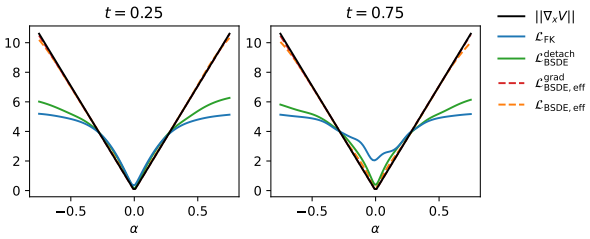


Figure 8. Approximation of the norm of the gradient $\|\nabla_x V\|$ of the solution V to the heat equation in Section 3.1 for $t \in \{0.25, 0.75\}$ and $x = (\alpha, \ldots, \alpha)^\top$ after training for $30k$ steps with batch size $K = 1024$. As expected, the stochastic integral, which is present in all losses except $\mathcal{L}_{\mathrm{FK}}$, induces better approximation of the gradient. Furthermore, explicitly modelling the gradient as in $\mathcal{L}^{\mathrm{grad}}_{\mathrm{BSDE,\,eff}}$ or back-propagating the derivative of the neural network as in $\mathcal{L}_{\mathrm{BSDE,\,eff}}$ also improves the approximation outside of the sampling interval for x, i.e. outside of $[-0.5, 0.5]^{50}$.

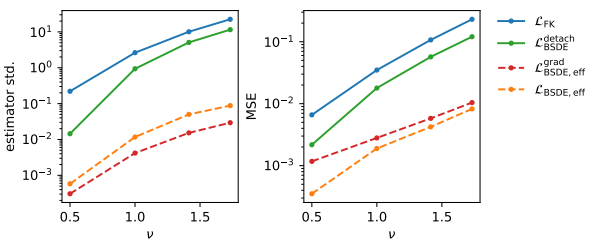


Figure 9. Estimator standard deviation and MSE after $30k$ gradient steps for the heat equation in Section 3.1 with batch size 1024 and varying diffusivities ν. While solving the PDE becomes more challenging for higher values of ν, our proposed methods consistently outperform the baseline.

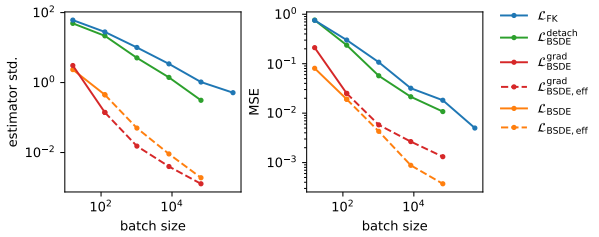


Figure 10. We observe performance differences similar to Figure 1 when solving the heat equation from Section 3.1 with a higher diffusivity of $\nu = \sqrt{3}$.

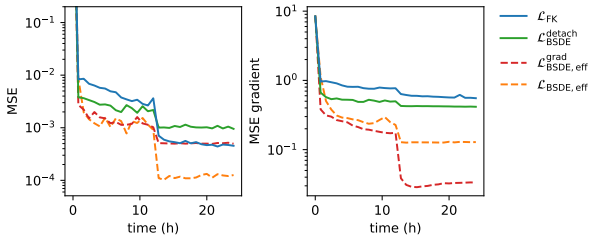


Figure 11. MSE of different losses as a function of the training time when solving the heat equation from Section 3.1 with batch size $K = 8192$. Although the time needed for computing and back-propagating the stochastic integral leads to significantly less gradient steps and thus samples of (ξ, τ, W) (see Figure 12), the loss $\mathcal{L}_{\mathrm{BSDE,\,eff}}$ still outperforms the loss $\mathcal{L}_{\mathrm{FK}}$.

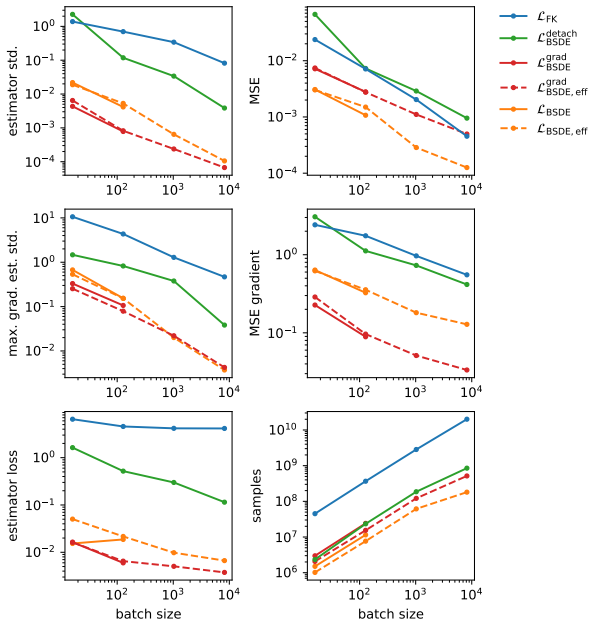


Figure 12. Scaling of estimator loss, standard deviation of the (gradient) estimator, and MSE when training for $24h$. We observe similar effects as in Figures 1 and 4, even though the number of samples (or, proportionally, the number of steps) is significantly higher for the loss $\mathcal{L}_{\mathrm{FK}}$. For batch sizes higher than $K = 8192$ a larger time budget is necessary.

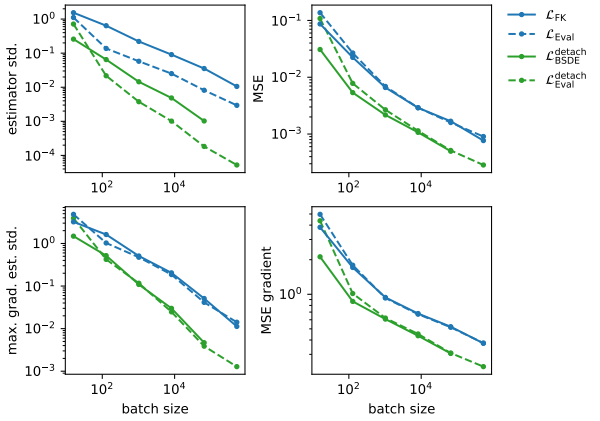


Figure 13. Performance and standard deviation of the losses $\mathcal{L}_{\mathrm{Eval}}$ and $\mathcal{L}^{\mathrm{detach}}_{\mathrm{Eval}}$ (detaching S_u in (7c) from the computational graph as in (30)) for the heat equation from Section 3.1 compared to their natural counterparts. Note that they perform significantly worse for small batch sizes, but similarly for large batch sizes – for the detached version one can allow for larger batch sizes, given a fixed memory budget. Note via a comparison to Figures 1 and 4 that the efficient versions of the considered losses are still much better.

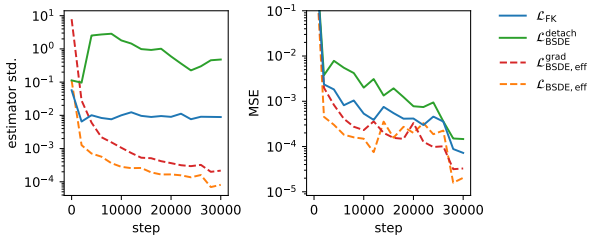


Figure 14. Estimator standard deviation and MSE as a function of the gradient steps when solving the Black–Scholes equation from Section 3.2 with batch size $K = 8192$. In case of the loss $\mathcal{L}^{\mathrm{detach}}_{\mathrm{BSDE}}$, a bad initial approximation actually leads to a variance-increasing effect of the stochastic integral. As the error in the gradient is not back-propagated to the network parameters, this deteriorates the convergence.

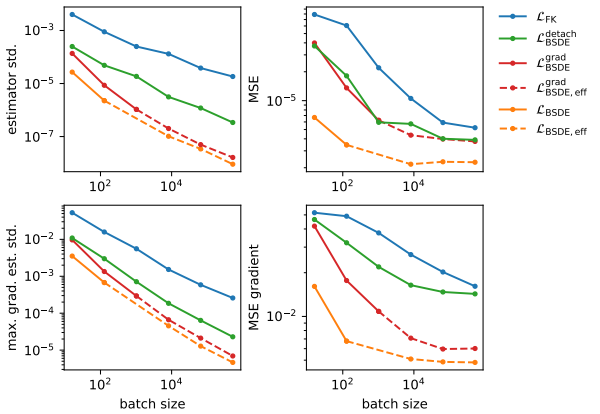


Figure 15. Similar to Figures 1, 4, and 5, our theoretical and empirical findings also hold in case of the HJB equation from Section 3.3.

G. Additional Material

The following Lemma details the relation of the HJB equation from Section 3.3 and a linear PDE of Kolmogorov type.

Lemma G.1 (Linearization of HJB equation)**.** *Let $\widetilde{V} \in C^{2,1}(\mathbb{R}^d \times [0,T], \mathbb{R})$ solve the Hamilton–Jacobi–Bellman equation in (45), then $V = \exp(-\widetilde{V})$ fulfills the linear PDE as stated in Theorem C.1 with $k = 0$ and $g = \exp(-\widetilde{g})$, i.e.*

$$\begin{aligned}(\partial_t + L - f(x,t))V(x,t) &= 0, \qquad && (x,t) \in \mathbb{R}^d \times [0,T),\\ V(x,T) &= g(x), && x \in \mathbb{R}^d.\end{aligned}$$

Subsequently, we get the representation

$$\widetilde{V}(x,t) = -\log \mathbb{E}\left[e^{-\mathcal{W}(X)} \middle| X_t = x\right] \tag{87}$$

with $\mathcal{W}(X) = \int_t^T f(X_s, s)\mathrm{d}s + \widetilde{g}(X_T)$.

Proof. We consider the transformation $V = \exp(-\widetilde{V})$ and for notational convenience omit the space and time dependencies of $\widetilde{V}$, f, b, and σ. We can compute

$$\begin{aligned}Le^{-\widetilde{V}} &= -b \cdot e^{-\widetilde{V}} \nabla_x \widetilde{V} - \frac{1}{2}\sum_{i,j=1}^d (\sigma\sigma^\top)_{ij} \partial_{x_i}\left(\partial_{x_j} e^{-\widetilde{V}} \widetilde{V}\right)\\
&= -e^{-\widetilde{V}}\Bigg(b \cdot \nabla_x \widetilde{V} + \frac{1}{2}\sum_{i,j=1}^d (\sigma\sigma^\top)_{ij}\partial_{x_i}\partial_{x_j}\widetilde{V}\\
&\qquad\qquad - \frac{1}{2}\sum_{i,j=1}^d (\sigma\sigma^\top)_{ij}\partial_{x_i}\widetilde{V}\partial_{x_j}\widetilde{V}\Bigg)\\
&= -e^{-\widetilde{V}}\Bigg(b \cdot \nabla_x \widetilde{V} + \frac{1}{2}\sum_{i,j=1}^d (\sigma\sigma^\top)_{ij}\partial_{x_i}\partial_{x_j}\widetilde{V}\\
&\qquad\qquad - \frac{1}{2}\sum_{i,j,k=1}^d \sigma_{ik}\sigma_{jk}\partial_{x_i}\widetilde{V}\partial_{x_j}\widetilde{V}\Bigg)\\
&= -e^{-\widetilde{V}}\Bigg(b \cdot \nabla_x \widetilde{V} + \frac{1}{2}\sum_{i,j=1}^d (\sigma\sigma^\top)_{ij}\partial_{x_i}\partial_{x_j}\widetilde{V}\\
&\qquad\qquad - \frac{1}{2}\sum_{k=1}^d \left(\sum_{i=1}^d \sigma_{ik}\partial_{x_i}\widetilde{V}\right)^2\Bigg)\\
&= -e^{-\widetilde{V}}\Big(L\widetilde{V} - \frac{1}{2}\|\sigma^\top \nabla_x \widetilde{V}\|^2\Big).\end{aligned}$$

The PDE in (69) therefore becomes

$$\begin{aligned}0 &= (\partial_t + L - f)e^{-\widetilde{V}} && \text{(88a)}\\
&= -e^{-\widetilde{V}}\Big((\partial_t + L)\widetilde{V} + f - \frac{1}{2}\|\sigma^\top \nabla_x \widetilde{V}\|^2\Big), && \text{(88b)}\end{aligned}$$

which is equivalent to the HJB equation in (45). □

For a discussion on the application of Lemma G.1 in the context of importance sampling of stochastic processes that are relevant in computational statistical physics and molecular dynamics we refer to Hartmann et al. (2017).

VI How Degenerate is the Parametrization of Neural Networks with the ReLU Activation Function?

Comments

Conference: Poster presentation at NeurIPS 2019.
E-Print: arXiv:1905.09803[cs.LG]

Contribution

Dennis Elbrächter and Julius Berner contributed equally to the concept, development, and writing. Philipp Grohs provided scientific advice and the initial idea for Theorem A.2.

Bibliographic Information

Berner, Julius, Dennis Elbrächter, and Philipp Grohs (2019). "How degenerate is the parametrization of neural networks with the ReLU activation function?" In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., pp. 7790–7801.

How degenerate is the parametrization of neural networks with the ReLU activation function?

Julius Berner
Faculty of Mathematics, University of Vienna
Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
julius.berner@univie.ac.at

Dennis Elbrächter
Faculty of Mathematics, University of Vienna
Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
dennis.elbraechter@univie.ac.at

Philipp Grohs
Faculty of Mathematics and Research Platform DataScience@UniVienna, University of Vienna
Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
philipp.grohs@univie.ac.at

Abstract

Neural network training is usually accomplished by solving a non-convex optimization problem using stochastic gradient descent. Although one optimizes over the networks parameters, the main loss function generally only depends on the realization of the neural network, i.e. the function it computes. Studying the optimization problem over the space of realizations opens up new ways to understand neural network training. In particular, usual loss functions like mean squared error and categorical cross entropy are convex on spaces of neural network realizations, which themselves are non-convex. Approximation capabilities of neural networks can be used to deal with the latter non-convexity, which allows us to establish that for sufficiently large networks local minima of a regularized optimization problem on the realization space are almost optimal. Note, however, that each realization has many different, possibly degenerate, parametrizations. In particular, a local minimum in the parametrization space needs not correspond to a local minimum in the realization space. To establish such a connection, inverse stability of the realization map is required, meaning that proximity of realizations must imply proximity of corresponding parametrizations. We present pathologies which prevent inverse stability in general, and, for shallow networks, proceed to establish a restricted space of parametrizations on which we have inverse stability w.r.t. to a Sobolev norm. Furthermore, we show that by optimizing over such restricted sets, it is still possible to learn any function which can be learned by optimization over unrestricted sets.

1 Introduction and Motivation

In recent years much effort has been invested into explaining and understanding the overwhelming success of deep learning based methods. On the theoretical side, impressive approximation capabilities of neural networks have been established [9, 10, 16, 20, 32, 33, 37, 39]. No less important are recent results on the generalization of neural networks, which deal with the question of how

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

well networks, trained on limited samples, perform on unseen data [2, 3, 5–7, 17, 29]. Last but not least, the optimization error, which quantifies how well a neural network can be trained by applying stochastic gradient descent to an optimization problem, has been analyzed in different scenarios [1, 11, 13, 22, 24, 25, 27, 38]. While there are many interesting approaches to the latter question, they tend to require very strong assumptions (e.g. (almost) linearity, convexity, or extreme over-parametrization). Thus a satisfying explanation for the success of stochastic gradient descent for a non-smooth, non-convex problem remains elusive.
In the present paper we intend to pave the way for a functional perspective on the optimization problem. This allows for new mathematical approaches towards understanding the training of neural networks, some of which are demonstrated in Section 1.2. To this end we examine degenerate parametrizations with undesirable properties in Section 2. These can be roughly classified as

C.1 unbalanced magnitudes of the parameters

C.2 weight vectors with the same direction

C.3 weight vectors with directly opposite directions.

Under conditions designed to avoid these degeneracies, Theorem 3.1 establishes inverse stability for shallow networks with ReLU activation function. This is accomplished by a refined analysis of the behavior of ReLU networks near a discontinuity of their derivative. Proposition 1.2 shows how inverse stability connects the loss surface of the parametrized minimization problem to the loss surface of the realization space problem. In Theorem 1.3 we showcase a novel result on almost optimality of local minima of the parametrized problem obtained by analyzing the realization space problem. Note that this approach of analyzing the loss surface is conceptually different from previous approaches as in [11, 18, 23, 30, 31, 36].

1.1 Inverse Stability of Neural Networks

We will focus on neural networks with the ReLU activation function $\rho(x) := x_+$, and adapt the mathematically convenient notation from [33], which distinguishes between the *parametrization* of a neural network and its *realization*. Let us define the set $\mathcal{A}_L$ of all network *architectures* with depth $L \in \mathbb{N}$, input dimension $d \in \mathbb{N}$, and output dimension $D \in \mathbb{N}$ by

$$\mathcal{A}_L := \{(N_0, \dots, N_L) \in \mathbb{N}^{L+1} \colon N_0 = d, N_L = D\}. \tag{1}$$

The architecture $N \in \mathcal{A}_L$ simply specifies the number of neurons N_l in each of the L layers. We can then define the space $\mathcal{P}_N$ of *parametrizations* with architecture $N \in \mathcal{A}_L$ as

$$\mathcal{P}_N := \prod_{\ell=1}^{L} \left(\mathbb{R}^{N_\ell \times N_{\ell-1}} \times \mathbb{R}^{N_\ell}\right), \tag{2}$$

the set $\mathcal{P} = \bigcup_{N \in \mathcal{A}_L} \mathcal{P}_N$ of all parametrizations with architecture in $\mathcal{A}_L$, and the *realization* map

$$\begin{aligned} \mathcal{R} \colon \mathcal{P} &\to C(\mathbb{R}^d, \mathbb{R}^D) \\ \Theta = ((A_\ell, b_\ell))_{\ell=1}^L &\mapsto \mathcal{R}(\Theta) := W_L \circ \rho \circ W_{L-1} \dots \rho \circ W_1, \end{aligned} \tag{3}$$

where $W_\ell(x) := A_\ell x + b_\ell$ and ρ is applied component-wise. We refer to A_ℓ and b_ℓ as the weights and biases in the ℓ-th layer.
Note that a parametrization $\Theta \in \Omega \subseteq \mathcal{P}$ uniquely induces a realization $\mathcal{R}(\Theta)$ in the realization space $\mathcal{R}(\Omega)$, while in general there can be multiple non-trivially different parametrizations with the same realization. To put it in mathematical terms, the realization map is not injective. Consider the basic counterexample

$$\Theta = \big((A_1, b_1), \dots, (A_{L-1}, b_{L-1}), (0, 0)\big) \quad \text{and} \quad \Gamma = \big((B_1, c_1), \dots, (B_{L-1}, c_{L-1}), (0, 0)\big) \tag{4}$$

from [34] where regardless of A_ℓ, B_ℓ, b_ℓ and c_ℓ both realizations coincide with $\mathcal{R}(\Theta) = \mathcal{R}(\Gamma) = 0$. However, it it is well-known that the realization map is locally Lipschitz continuous, meaning that close[1] parametrizations in $\mathcal{P}_N$ induce realizations which are close in the uniform norm on compact

[1] On the finite dimensional vector space $\mathcal{P}_N$ all norms are equivalent and we take w.l.o.g. the maximum norm $\|\Theta\|_\infty$, i.e. the maximum of the absolute values of the entries of the A_ℓ and b_ℓ.

sets, see e.g. [2, Lemma 14.6], [7, Theorem 4.2], and [34, Proposition 5.1].
We will shed light upon the inverse question. Given realizations $\mathcal{R}(\Gamma)$ and $\mathcal{R}(\Theta)$ that are close, do the parametrizations Γ and Θ have to be close? In an abstract setting we measure the proximity of realizations in the norm $\|\cdot\|$ of a Banach space $\mathcal{B}$ with $\mathcal{R}(\mathcal{P}) \subseteq \mathcal{B}$, while concrete Banach spaces of interest will be specified later. In view of the above counterexample we will, at the very least, need to allow for the reparametrization of one of the networks, i.e. we arrive at the following question.

> Given $\mathcal{R}(\Gamma)$ and $\mathcal{R}(\Theta)$ that are close, does there exist a parametrization Φ with $\mathcal{R}(\Phi) = \mathcal{R}(\Theta)$ such that Γ and Φ are close?

As we will see in Section 2, this question is fundamentally connected to understanding the redundancies and degeneracies of the way that neural networks are parametrized. By suitable regularization, i.e. considering a subspace $\Omega \subseteq \mathcal{P}_N$ of parametrizations, we can avoid these pathologies and establish a positive answer to the question above. For such a property the term *inverse stability* was introduced in [34], which constitutes the only other research conducted in this area, as far as we are aware.

Definition 1.1 (Inverse stability)**.** *Let $s, \alpha > 0$, $N \in \mathcal{A}_L$, and $\Omega \subseteq \mathcal{P}_N$. We say that the realization map is (s, α) inverse stable on Ω w.r.t. $\|\cdot\|$, if for all $\Gamma \in \Omega$ and $g \in \mathcal{R}(\Omega)$ there exists $\Phi \in \Omega$ with*

$$\mathcal{R}(\Phi) = g \quad \textit{and} \quad \|\Phi - \Gamma\|_\infty \leq s\|g - \mathcal{R}(\Gamma)\|^\alpha. \tag{5}$$

In Section 2 we will see why inverse stability fails w.r.t. the uniform norm. Therefore, we consider a norm which takes into account not only the maximum error of the function values but also of the gradients. In mathematical terms, we make use of the Sobolev norm $\|\cdot\|_{W^{1,\infty}(U)}$ (on some domain $U \subseteq \mathbb{R}^d$) defined for every (locally) Lipschitz continuous function $g\colon \mathbb{R}^d \to \mathbb{R}^D$ by $\|g\|_{W^{1,\infty}(U)} := \max\{\|g\|_{L^\infty(U)}, |g|_{W^{1,\infty}(U)}\}$ with the Sobolev semi-norm $|\cdot|_{W^{1,\infty}(U)}$ given by

$$|g|_{W^{1,\infty}(U)} := \|Dg\|_{L^\infty(U)} = \operatorname*{ess\,sup}_{x \in U} \|Dg(x)\|_\infty. \tag{6}$$

See [15] for further information on Sobolev norms, and [8] for further information on the derivative of ReLU networks.

1.2 Implications of inverse stability for neural network optimization

We proceed by demonstrating how inverse stability opens up new perspectives on the optimization problem which arises in neural network training. Specifically, consider a loss function $\mathcal{L}\colon C(\mathbb{R}^d, \mathbb{R}^D) \to [0, \infty)$ on the space of continuous functions. For illustration, we take the commonly used mean squared error (MSE) which, for training data $((x^i, y^i))_{i=1}^n \in (\mathbb{R}^d \times \mathbb{R}^D)^n$, is given by

$$\mathcal{L}(g) = \tfrac{1}{n} \sum_{i=1}^n \|g(x^i) - y^i\|_2^2, \quad \text{for } g \in C(\mathbb{R}^d, \mathbb{R}^D). \tag{7}$$

Typically, the optimization problem is solved over some subspace of parametrizations $\Omega \subseteq \mathcal{P}_N$, i.e.

$$\min_{\Gamma \in \Omega} \mathcal{L}(\mathcal{R}(\Gamma)) = \min_{\Gamma \in \Omega} \tfrac{1}{n} \sum_{i=1}^n \|\mathcal{R}(\Gamma)(x^i) - y^i\|_2^2. \tag{8}$$

From an abstract point of view, by writing $g = \mathcal{R}(\Gamma) \in \mathcal{R}(\Omega)$, this is equivalent to the corresponding optimization problem over the space of realizations $\mathcal{R}(\Omega)$, i.e.

$$\min_{g \in \mathcal{R}(\Omega)} \mathcal{L}(g) = \min_{g \in \mathcal{R}(\Omega)} \tfrac{1}{n} \sum_{i=1}^n \|g(x^i) - y^i\|_2^2. \tag{9}$$

However, the loss landscape of the optimization problem (8) is only properly connected to the loss landscape of the optimization problem (9) if the realization map is inverse stable on Ω. Otherwise a realization $g \in \mathcal{R}(\mathcal{P}_N)$ can be arbitrarily close to a global minimum in the realization space but every parametrization Φ with $\mathcal{R}(\Phi) = g$ is far away from the corresponding global minimum in the parametrization space. Moreover, local minima of (8) in the parametrization space must correspond to local minima of (9) in the realization space if and only if we have inverse stability.

Proposition 1.2 (Parametrization minimum ⇒ realization minimum)**.** *Let $N \in \mathcal{A}_L$, $\Omega \subseteq \mathcal{P}_N$ and let the realization map be (s, α) inverse stable on Ω w.r.t. $\|\cdot\|$. Let $\Gamma_* \in \Omega$ be a local minimum of $\mathcal{L} \circ \mathcal{R}$ on Ω with radius $r > 0$, i.e. for all $\Phi \in \Omega$ with $\|\Phi - \Gamma_*\|_\infty \leq r$ it holds that*

$$\mathcal{L}(\mathcal{R}(\Gamma_*)) \leq \mathcal{L}(\mathcal{R}(\Phi)). \tag{10}$$

Then $\mathcal{R}(\Gamma_*)$ is a local minimum of $\mathcal{L}$ on $\mathcal{R}(\Omega)$ with radius $(\frac{r}{s})^{1/\alpha}$, i.e. for all $g \in \mathcal{R}(\Omega)$ with $\|g - \mathcal{R}(\Gamma_*)\| \leq (\frac{r}{s})^{1/\alpha}$ it holds that

$$\mathcal{L}(\mathcal{R}(\Gamma_*)) \leq \mathcal{L}(g). \tag{11}$$

See Appendix A.1.2 for a proof and Example A.1 for a counterexample in the case that inverse stability is not given. Note that in (9) we consider a problem with convex loss function but non-convex feasible set, see [34, Section 3.2]. This opens up new avenues of investigation using tools from functional analysis and allows utilizing recent results [19, 34] exploring the topological properties of neural network realization spaces.

As a concrete demonstration we provide with Theorem A.2 a strong result obtained on the realization space, which estimates the quality of a local minimum based on its radius and the approximation capabilities of the chosen architecture for a class of functions S. Specifically let $C > 0$, let $\Lambda \colon \mathcal{B} \to [0, \infty)$ be a quasi-convex regularizer, and define

$$S := \{f \in \mathcal{B} \colon \Lambda(f) \leq C\}. \tag{12}$$

We denote the sets of regularized parametrizations by

$$\Omega_N := \{\Phi \in \mathcal{P}_N \colon \Lambda(\mathcal{R}(\Phi)) \leq C\} \tag{13}$$

and assume that the loss function $\mathcal{L}$ is convex and c-Lipschitz continuous on S. Note that virtually all relevant loss functions are convex and locally Lipschitz continuous on $C(\mathbb{R}^d, \mathbb{R}^D)$. Employing Proposition 1.2, inverse stability can then be used to derive the following result for the practically relevant parametrized problem, showing that for sufficiently large architectures local minima of a regularized neural network optimization problem are almost optimal.

Theorem 1.3 (Almost optimality of local parameter minima)**.** *Assume that S is compact in the $\|\cdot\|$-closure of $\mathcal{R}(\mathcal{P})$ and that for every $N \in \mathcal{A}_L$ the realization map is (s, α) inverse stable on Ω_N w.r.t. $\|\cdot\|$. Then for all $\varepsilon, r > 0$ there exists $n(\varepsilon, r) \in \mathcal{A}_L$ such that for every $N \in \mathcal{A}_L$ with $N_1 \geq n_1(\varepsilon, r), \ldots, N_{L-1} \geq n_{L-1}(\varepsilon, r)$ the following holds:*
Every local minimum Γ_* with radius at least r of $\min_{\Gamma \in \Omega_N} \mathcal{L}(\mathcal{R}(\Gamma))$ satisfies

$$\mathcal{L}(\mathcal{R}(\Gamma_*)) \leq \min_{\Gamma \in \Omega_N} \mathcal{L}(\mathcal{R}(\Gamma)) + \varepsilon. \tag{14}$$

See Appendix A.1.2 for a proof and note that here it is important to have an inverse stability result, where the parameters (s, α) do not depend on the size of the architecture, which we achieve for $L = 2$ and $\mathcal{B} = W^{1,\infty}$. Suitable Λ would be Besov norms which constitute a common regularizer in image and signal processing. Moreover, note that the required size of the architecture in Theorem 1.3 can be quantified, if one has approximation rates for S. In particular, this approach allows the use of approximation results in order to explain the success of neural network optimization and enables a combined study of these two aspects, which, to the best of our knowledge, has not been done before. Unlike in recent literature, our result needs no assumptions on the sample set (incorporated in the loss function, see (7)), in particular we do not require "overparametrization" with respect to the sample size. Here the required size of the architecture only depends on the complexity of S, i.e. the class of functions one wants to approximate, the radius of the local minima of interest, the Lipschitz constant of the loss function, and the parameters of the inverse stability.

In the following we restrict ourselves to two-layer ReLU networks without biases, where we present a proof for $(4, 1/2)$ inverse stability w.r.t. the Sobolev semi-norm on a suitably regularized space of parametrizations. Both the regularizations as well as the stronger norm (compared to the uniform norm) will shown to be necessary in Section 2. We now present, in an informal way, a collection of our main results. A short proof making the connection to the formal results can be found in Appendix A.1.2.

Corollary 1.4 (Inverse stability and implications - colloquial)**.** *Suppose we are given data $((x^i, y^i))_{i=1}^n \in (\mathbb{R}^d \times \mathbb{R}^D)^n$ and want to solve a typical minimization problem for ReLU networks with shallow architecture $N = (d, N_1, D)$, i.e.*

$$\min_{\Gamma \in \mathcal{P}_N} \frac{1}{n} \sum_{i=1}^n \|\mathcal{R}(\Gamma)(x^i) - y^i)\|_2^2. \tag{15}$$

First we augment the architecture to $\tilde{N} = (d+2, N_1+1, D)$, while omitting the biases, and augment the samples to $\tilde{x}^i = (x_1^i, \dots, x_d^i, 1, -1)$. Moreover, we assume that the parametrizations

$$\Phi = \big(([a_1|\dots|a_{N_1+1}]^T, 0), ([c_1|\dots|c_{N_1+1}], 0)\big) \in \Omega \subseteq \mathcal{P}_{\tilde{N}} \tag{16}$$

are regularized such that

- *C.1 the network is balanced, i.e. $\|a_i\|_\infty = \|c_i\|_\infty$,*
- *C.2 no non-zero weight vectors in the first layer are redundant, i.e. $a_i \nparallel a_j$,*
- *C.3 the last two coordinates of each weight vector a_i are strictly positive.*

Then for the new minimization problem

$$\min_{\Phi\in\Omega} \tfrac{1}{n} \sum_{i=1}^{n} \|\mathcal{R}(\Phi)(\tilde{x}^i) - y^i\|_2^2 \tag{17}$$

the following holds:

1. *If Φ_* is a local minimum of* (17) *with radius r, then $\mathcal{R}(\Phi_*)$ is a local minimum of $\min_{g\in\mathcal{R}(\Omega)} \frac{1}{n}\sum_{i=1}^n \|g(\tilde{x}^i) - y^i\|_2^2$ with radius at least $\frac{r^2}{16}$ w.r.t. $|\cdot|_{W^{1,\infty}}$.*
2. *The global minimum of* (17) *is at least as good as the global minimum of* (15)*, i.e.*

$$\min_{\Phi\in\Omega} \tfrac{1}{n} \sum_{i=1}^{n} \|\mathcal{R}(\Phi)(\tilde{x}^i) - y^i\|_2^2 \leq \min_{\Gamma\in\mathcal{P}_N} \tfrac{1}{n} \sum_{i=1}^{n} \|\mathcal{R}(\Gamma)(x^i) - y^i\|_2^2. \tag{18}$$

3. *By further regularizing* (17) *in the sense of Theorem 1.3, we can estimate the quality of its local minima.*

This argument is not limited to the MSE loss function but works for any loss function based on evaluating the realization. The omission of bias weights is standard in neural network optimization literature [11, 13, 22, 24]. While this severely limits the functions that can be realized with a given architecture, it is sufficient to augment the problem by one dimension in order to recover the full range of functions that can be learned [1]. Here we augment by two dimensions, so that the third regularization condition C.3 can be fulfilled without loosing range. Moreover, note that, for simplicity of presentation, the regularization assumptions stated above are stricter than necessary and possible relaxations are discussed in Section 3.

2 Obstacles to inverse stability - degeneracies of ReLU parametrizations

In the remainder of this paper we focus on shallow ReLU networks without biases and define the corresponding space of parametrizations with architecture $N = (d, m, D)$ as $\mathcal{N}_N := \mathbb{R}^{m\times d} \times \mathbb{R}^{D\times m}$. The realization map[2] $\mathcal{R}$ is, for every $\Theta = (A, C) = \big([a_1|\dots|a_m]^T, [c_1|\dots|c_m]\big) \in \mathcal{N}_N$, given by

$$\mathbb{R}^d \ni x \mapsto \mathcal{R}(\Theta)(x) = C\rho(Ax) = \sum_{i=1}^{m} c_i \rho(\langle a_i, x\rangle). \tag{19}$$

Note that each function $x \mapsto c_i\rho(\langle a_i, x\rangle)$ represents a so-called ridge function which is zero on the half-space $\{x \in \mathbb{R}^d \colon \langle a_i, x\rangle \leq 0\}$ and linear with constant derivative $c_i a_i^T \in \mathbb{R}^D \times \mathbb{R}^d$ on the other half-space. Thus, the a_i are the normal vectors of the separating hyperplanes $\{x \in \mathbb{R}^d \colon \langle a_i, x\rangle = 0\}$ and consequently we refer to the weight vectors a_i also as the directions of Θ. Moreover, for $\Theta \in \mathcal{N}_N$ it holds that $\mathcal{R}(\Theta)(0) = 0$ and, as long as the domain of interest $U \subseteq \mathbb{R}^d$ contains the origin, the Sobolev norm $\|\cdot\|_{W^{1,\infty}(U)}$ is equivalent to its semi-norm, since

$$\|\mathcal{R}(\Theta)\|_{L^\infty(U)} \leq \sqrt{d}\,\operatorname{diam}(U) |\mathcal{R}(\Theta)|_{W^{1,\infty}}, \tag{20}$$

[2] This is a slight abuse of notation, justified by the the fact that $\mathcal{R}$ acts the same on $\mathcal{P}_N$ with zero biases b_1, b_2 and weights $A_1 = A$ and $A_2 = C$.

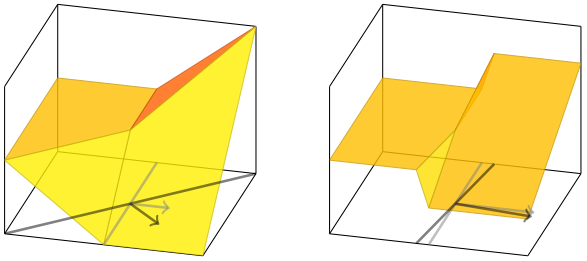


Figure 1: The figure shows g_k for $k = 1, 2$.

see also inequalities of Poincaré-Friedrichs type [14, Subsection 5.8.1]. Therefore, in the rest of the paper we will only consider the Sobolev semi-norm[3]

$$|\mathcal{R}(\Theta)|_{W^{1,\infty}(U)} = \operatorname*{ess\,sup}_{x\in U} \Big\| \sum_{i\in[m]\colon \langle a_i, x\rangle > 0} c_i a_i^T \Big\|_\infty. \tag{21}$$

In (21) one can see that in our setting $|\cdot|_{W^{1,\infty}(U)}$ is independent of U (as long as U contains a neighbourhood of the origin) and will thus be abbreviated by $|\cdot|_{W^{1,\infty}}$.

2.1 Failure of inverse stability w.r.t. uniform norm

All proofs for this section can be found in Appendix A.2.2. We start by showing that inverse stability fails w.r.t. the uniform norm. This example is adapted from [34, Theorem 5.2] and represents, to the best of our knowledge, the only degeneracy which has already been observed before.

Example 2.1 (Failure due to exploding gradient)**.** *Let* $\Gamma := (0,0) \in \mathcal{N}_{(2,2,1)}$ *and* $g_k \in \mathcal{R}(\mathcal{N}_{(2,2,1)})$ *be given by (see Figure 1)*

$$g_k(x) := k\rho(\langle (k,0), x\rangle) - k\rho(\langle (k, -\tfrac{1}{k^2}), x\rangle), \quad k \in \mathbb{N}. \tag{22}$$

Then for every sequence $(\Phi_k)_{k\in\mathbb{N}} \subseteq \mathcal{N}_{(2,2,1)}$ *with* $\mathcal{R}(\Phi_k) = g_k$ *it holds that*

$$\lim_{k\to\infty} \|\mathcal{R}(\Phi_k) - \mathcal{R}(\Gamma)\|_{L^\infty((-1,1)^2)} = 0 \quad \textit{and} \quad \lim_{k\to\infty} \|\Phi_k - \Gamma\|_\infty = \infty. \tag{23}$$

In particular, note that inverse stability fails here even for a non-degenerate parametrization of the zero function $\Gamma = (0,0)$. However, for this type of counterexample the magnitude of the gradient of $\mathcal{R}(\Phi_k)$ needs to go to infinity, which is our motivation for looking at inverse stability w.r.t. $|\cdot|_{W^{1,\infty}}$.

2.2 Failure of inverse stability w.r.t. Sobolev norm

In this section we present four degenerate cases where inverse stability fails w.r.t. $|\cdot|_{W^{1,\infty}}$. This collection of counterexamples is complete in the sense that we can establish inverse stability under assumptions which are designed to exclude these four pathologies.

Example 2.2 (Failure due to complete unbalancedness)**.** *Let* $r > 0$, $\Gamma := \big((r,0),0\big) \in \mathcal{N}_{(2,1,1)}$ *and* $g_k \in \mathcal{R}(\mathcal{N}_{(2,1,1)})$ *be given by (see Figure 2)*

$$g_k(x) = \tfrac{1}{k}\rho(\langle (0,1), x\rangle), \quad k \in \mathbb{N}. \tag{24}$$

Then for every $k \in \mathbb{N}$ *and* $\Phi_k \in \mathcal{N}_{(2,1,1)}$ *with* $\mathcal{R}(\Phi_k) = g_k$ *it holds that*

$$|\mathcal{R}(\Phi_k) - \mathcal{R}(\Gamma)|_{W^{1,\infty}} = \tfrac{1}{k} \quad \textit{and} \quad \|\Phi_k - \Gamma\|_\infty \geq r. \tag{25}$$

This is a very simple example of a degenerate parametrization of the zero function, since $\mathcal{R}(\Gamma) = 0$ regardless of choice of r. The issue here is that we can have a weight pair, i.e. $((r,0),0)$, where the product is independent of the value of one of the parameters. Note that in Example A.4 one can see a slightly more subtle version of this pathology by considering $\Gamma_k := \big((k,0), \frac{1}{k^2}\big) \in \mathcal{N}_{(2,1,1)}$ instead. In that case one could still get an inverse stability estimate for each fixed k; the parameters of inverse

[3] For $m \in \mathbb{N}$ we abbreviate $[m] := \{1, \dots, m\}$.

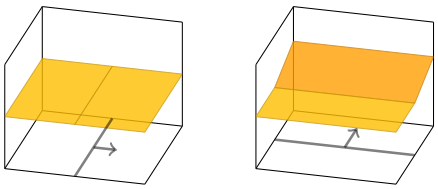


Figure 2: Shows $\mathcal{R}(\Gamma)$ ($r = 0.5$) and g_3.

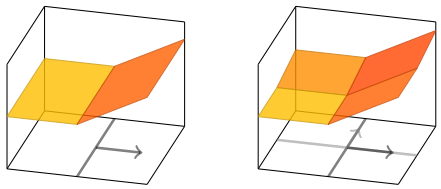


Figure 3: Shows $\mathcal{R}(\Gamma)$ and g_2.

stability (s, α) would however deteriorate with increasing k. In particular this demonstrates the need for some sort of balancedness of the parametrization, i.e. control over $\|c_i\|_\infty$ and $\|a_i\|_\infty$ individually relative to $\|c_i\|_\infty \|a_i\|_\infty$.
Inverse stability is also prevented by redundant directions as the following example illustrates.

Example 2.3 (Failure due to redundant directions)**.** *Let*

$$\Gamma := \left(\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, (1, 1) \right) \in \mathcal{N}_{(2,2,1)} \tag{26}$$

and $g_k \in \mathcal{R}(\mathcal{N}_{(2,2,1)})$ be given by (see Figure 3)

$$g_k(x) := 2\rho(\langle (1, 0), x \rangle) + \tfrac{1}{k}\rho(\langle (0, 1), x \rangle), \quad k \in \mathbb{N}. \tag{27}$$

Then for every $k \in \mathbb{N}$ and $\Phi_k \in \mathcal{N}_{(2,2,1)}$ with $\mathcal{R}(\Phi_k) = g_k$ it holds that

$$|\mathcal{R}(\Phi_k) - \mathcal{R}(\Gamma)|_{W^{1,\infty}} = \tfrac{1}{k} \quad \textit{and} \quad \|\Phi_k - \Gamma\|_\infty \geq 1. \tag{28}$$

The next example shows that not only redundant weight vectors can cause issues, but also weight vectors of opposite direction, as they would allow for a (balanced) degenerate parametrization of the zero function.

Example 2.4 (Failure due to opposite weight vectors 1)**.** *Let $a_i \in \mathbb{R}^d$, $i \in [m]$, be pairwise linearly independent with $\|a_i\|_\infty = 1$ and $\sum_{i=1}^m a_i = 0$. We define*

$$\Gamma := \big([a_1| \dots |a_m| - a_1| \dots | - a_m]^T, (1, \dots, 1, -1, \dots, -1) \big) \in \mathcal{N}_{(d,2m,1)}. \tag{29}$$

Now let $v \in \mathbb{R}^d$ with $\|v\|_\infty = 1$ be linearly independent to each a_i, $i \in [m]$, and let $g_k \in \mathcal{R}(\mathcal{N}_{(d,2m,1)})$ be given by (see Figure 4)

$$g_k(x) = \tfrac{1}{k}\rho(\langle v, x \rangle), \quad k \in \mathbb{N}. \tag{30}$$

Then there exists a constant $C > 0$ such that for every $k \in \mathbb{N}$ and every $\Phi_k \in \mathcal{N}_{(d,2m,1)}$ with $\mathcal{R}(\Phi_k) = g_k$ it holds that

$$|\mathcal{R}(\Phi_k) - \mathcal{R}(\Gamma)|_{W^{1,\infty}} = \tfrac{1}{k} \quad \textit{and} \quad \|\Phi_k - \Gamma\|_\infty \geq C. \tag{31}$$

Thus we will need an assumption which prevents each individual Γ in our restricted set from having pairwise linearly dependent weight vectors, i.e. coinciding hyperplanes of non-differentiability. This, however, does not suffice as is demonstrated by the next example, which shows that the relation between the hyperplanes of the two realizations matters.

Example 2.5 (Failure due to opposite weight vectors 2)**.** *We define the weight vectors*

$$a_1^k = (k, k, \tfrac{1}{k}), \quad a_2^k = (-k, k, \tfrac{1}{k}), \quad a_3^k = (0, -\sqrt{2}k, \tfrac{1}{\sqrt{2}k}), \quad c^k = (k, k, \sqrt{2}k) \tag{32}$$

and consider the parametrizations (see Figure 5)

$$\Gamma_k := \left(\big[-a_1^k| - a_2^k| - a_3^k \big]^T, c^k \right) \in \mathcal{N}_{(3,3,1)}, \quad \Theta_k := \left(\big[a_1^k | a_2^k | a_3^k \big]^T, c^k \right) \in \mathcal{N}_{(3,3,1)}. \tag{33}$$

Then for every $k \in \mathbb{N}$ and every $\Phi_k \in \mathcal{N}_{(3,3,1)}$ with $\mathcal{R}(\Phi_k) = \mathcal{R}(\Theta_k)$ it holds that

$$|\mathcal{R}(\Phi_k) - \mathcal{R}(\Gamma_k)|_{W^{1,\infty}} = 3 \quad \textit{and} \quad \|\Phi_k - \Gamma_k\|_\infty \geq k. \tag{34}$$

Note that Γ and Θ need to have multiple exactly opposite weight vectors which add to something small (compared to the size of the individual vectors), but not zero, since otherwise reparametrization would be possible (see Lemma A.5).

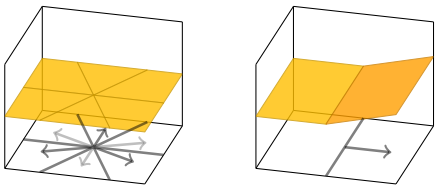


Figure 4: Shows $\mathcal{R}(\Gamma)$ and g_3 ($a_1 = (1, -\frac{1}{2})$, $a_2 = (-1, -\frac{1}{2})$, $a_3 = (0, 1)$, $v = (1, 0)$).

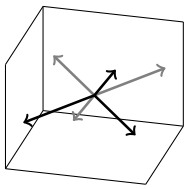


Figure 5: Shows the weight vectors of Θ_2 (grey) and Γ_2 (black).

3 Inverse stability for two-layer ReLU Networks

We now establish an inverse stability result using assumptions designed to exclude the pathologies from the previous section. First we present a rather technical theorem for output dimension one which considers a parametrization Γ in the unrestricted parametrization space $\mathcal{N}_N$ and a function g in the the corresponding function space $\mathcal{R}(\mathcal{N}_N)$. The aim is to use assumptions which are as weak as possible, while allowing us to find a parametrization Φ of g, whose distance to Γ can be bounded relative to $|g - \mathcal{R}(\Gamma)|_{W^{1,\infty}}$. We then continue by defining a restricted parametrization space $\mathcal{N}_N^*$, for which we get uniform inverse stability (meaning that we get the same estimate for every $\Gamma \in \mathcal{N}_N^*$).

Theorem 3.1 (Inverse stability at $\Gamma \in \mathcal{N}_N$)**.** *Let $d, m \in \mathbb{N}$, $N := (d, m, 1)$, $\beta \in [0, \infty)$, let $\Gamma = \left(\left[a_1^\Gamma \middle| \dots \middle| a_m^\Gamma \right]^T, c^\Gamma \right) \in \mathcal{N}_N$, $g \in \mathcal{R}(\mathcal{N}_N)$, and let $I^\Gamma := \{i \in [m] \colon a_i^\Gamma \neq 0\}$.*
Assume that the following conditions are satisfied:

- *C.1 It holds for all $i \in [m]$ with $\|c_i^\Gamma a_i^\Gamma\|_\infty \leq 2|g - \mathcal{R}(\Gamma)|_{W^{1,\infty}}$ that $|c_i^\Gamma|, \|a_i^\Gamma\|_\infty \leq \beta$.*
- *C.2 It holds for all $i, j \in I^\Gamma$ with $i \neq j$ that $\frac{a_j^\Gamma}{\|a_j^\Gamma\|_\infty} \neq \frac{a_i^\Gamma}{\|a_i^\Gamma\|_\infty}$.*
- *C.3 There exists a parametrization $\Theta = \left(\left[a_1^\Theta \middle| \dots \middle| a_m^\Theta \right]^T, c^\Theta \right) \in \mathcal{N}_N$ such that $\mathcal{R}(\Theta) = g$ and*
  - *(a) it holds for all $i, j \in I^\Gamma$ with $i \neq j$ that $\frac{a_j^\Gamma}{\|a_j^\Gamma\|_\infty} \neq -\frac{a_i^\Gamma}{\|a_i^\Gamma\|_\infty}$ and for all $i, j \in I^\Theta$ with $i \neq j$ that $\frac{a_j^\Theta}{\|a_j^\Theta\|_\infty} \neq -\frac{a_i^\Theta}{\|a_i^\Theta\|_\infty}$,*
  - *(b) it holds for all $i \in I^\Gamma$, $j \in I^\Theta$ that $\frac{a_i^\Gamma}{\|a_i^\Gamma\|_\infty} \neq -\frac{a_j^\Theta}{\|a_j^\Theta\|_\infty}$*

  where $I^\Theta := \{i \in [m] \colon a_i^\Theta \neq 0\}$.

Then there exists a parametrization $\Phi \in \mathcal{N}_N$ with

$$\mathcal{R}(\Phi) = g \quad \text{and} \quad \|\Phi - \Gamma\|_\infty \leq \beta + 2|g - \mathcal{R}(\Gamma)|_{W^{1,\infty}}^{\frac{1}{2}}. \tag{35}$$

The proof can be found in Appendix A.3.2. Note that each of the conditions in the theorem above corresponds directly to one of the pathologies in Section 2.2. Condition C.1, which deals with unbalancedness, only imposes an restriction on the weight pairs whose product is small compared to the distance of $\mathcal{R}(\Gamma)$ and g. As can be guessed from Example 2.2 and seen in the proof of Theorem 3.1, such a balancedness assumption is in fact only needed to deal with degenerate cases, where $\mathcal{R}(\Gamma)$ and g have parts with mismatching directions of negligible magnitude. Otherwise a matching reparametrization is always possible. Note that a balanced Γ (i.e. $|c_i^\Gamma| = \|a_i^\Gamma\|_\infty$) satisfies Condition C.1 with $\beta = (2|g - \mathcal{R}(\Gamma)|_{W^{1,\infty}})^{1/2}$.
It is also possible to relax the balancedness assumption by only requiring $|c_i^\Gamma|$ and $\|\Gamma_i\|_\infty$ to be close to $\|c_i^\Gamma a_i^\Gamma\|_\infty^{1/2}$, which would still give a similar estimate but with a worse exponent. In order to see that requiring balancedness does not restrict the space of realizations, observe that the ReLU is positively homogeneous (i.e. $\rho(\lambda x) = \lambda \rho(x)$ for all $\lambda \geq 0$, $x \in \mathbb{R}$). Thus balancedness can always be achieved simply by rescaling.
Condition C.2 requires Γ to have no redundant directions, the necessity of which is demonstrated by Example 2.3. Note that prohibiting redundant directions does not restrict the space of realizations,

see (87) in the appendix for details. From a practical point of view, enforcing this condition could be achieved by a regularization term using a barrier function. Alternatively on could employ a non-standard approach of combining such redundant neurons by changing one of them according to (87) and either setting the other one to zero or removing it entirely[4].
From a theoretical perspective the first two conditions are rather mild, in the sense that they only restrict the space of parametrizations and not the corresponding space of realizations. Specifically we can define the restricted parametrization space

$$\mathcal{N}'_{(d,m,D)} := \{\Gamma \in \mathcal{N}_{(d,m,D)} \colon \|c_i^\Gamma\|_\infty = \|a_i^\Gamma\|_\infty \text{ for all } i \in [m] \text{ and } \Gamma \text{ satisfies C.2}\} \tag{36}$$

for which we have $\mathcal{R}(\mathcal{N}'_N) = \mathcal{R}(\mathcal{N}_N)$. Note that the above definition as well as the following definition and theorem are for networks with arbitrary output dimensions, as the balancedness condition makes this extension rather straightforward.
In order to satisfy Conditions C.3a and C.3b we need to restrict the parametrization space in a way which also restricts the corresponding space of realizations. One possibility to do so is the following approach, which also incorporates the previous restrictions as well as the transition to networks without biases.

Definition 3.2 (Restricted parametrization space)**.** *Let $N = (d, m, D) \in \mathbb{N}^3$. We define*

$$\mathcal{N}_N^* := \left\{\Gamma \in \mathcal{N}'_N \colon (a_i^\Gamma)_{d-1}, (a_i^\Gamma)_d > 0 \text{ for all } i \in [m]\right\}. \tag{37}$$

While we no longer have $\mathcal{R}(\mathcal{N}_N^*) = \mathcal{R}(\mathcal{N}_N)$, Lemma A.6 shows that for every $\Theta \in \mathcal{P}_{(d,m,D)}$ there exists $\Gamma \in \mathcal{N}^*_{(d+2,m+1,D)}$ such that for all $x \in \mathbb{R}^d$ it holds that

$$\mathcal{R}(\Gamma)(x_1, \dots, x_d, 1, -1) = \mathcal{R}(\Theta)(x_1, \dots, x_d). \tag{38}$$

In particular, this means that for any optimization problem over an unrestricted parametrization space $\mathcal{P}_{(d,m,D)}$, there is a corresponding optimization problem over the parametrization space $\mathcal{N}^*_{(d+2,m+1,D)}$ whose solution is at least as good (see Corollary 1.4). Our main result now states that for such a restricted parametrization space we have uniform $(4, 1/2)$ inverse stability w.r.t. $|\cdot|_{W^{1,\infty}}$, a proof of which can be found in Appendix A.3.2.

Theorem 3.3 (Inverse stability on $\mathcal{N}_N^*$)**.** *Let $N \in \mathbb{N}^3$. For all $\Gamma \in \mathcal{N}_N^*$ and $g \in \mathcal{R}(\mathcal{N}_N^*)$ there exists a parametrization $\Phi \in \mathcal{N}_N^*$ with*

$$\mathcal{R}(\Phi) = g \quad \textit{and} \quad \|\Phi - \Gamma\|_\infty \leq 4|g - \mathcal{R}(\Gamma)|^{\frac{1}{2}}_{W^{1,\infty}}. \tag{39}$$

4 Outlook

This contribution investigates the potential insights which may be gained from studying the optimization problem over the space of realizations, as well as the difficulties encountered when trying to connect it to the parametrized problem. While Theorem 1.3 and Theorem 3.3 offer some compelling preliminary answers, there are multiple ways in which they can be extended.
To obtain our inverse stability result for shallow ReLU networks we studied sums of ridge functions. Extending this result to deep ReLU networks requires understanding their behaviour under composition. In particular, we have ridge functions which vanish on some half space, i.e. colloquially speaking each neuron may "discard half the information" it receives from the previous layer. This introduces a new type of degeneracy, which one will have to deal with.
Another interesting direction is an extension to inverse stability w.r.t. some weaker norm like $\|\cdot\|_{L^\infty}$ or a fractional Sobolev norm under stronger restrictions on the space of parametrizations (see Lemma A.7 for a simple approach using very strong restrictions).
Lastly, note that Theorem 1.3 is not specific to the ReLU activation function and thus also incentivizes the study of inverse stability for any other activation function.
From an applied point of view, Conditions C.1-C.3 motivate the implementation of corresponding regularization (i.e. penalizing unbalancedness and redundancy in the sense of parallel weight vectors) in state-of-the-art networks, in order to explore whether preventing inverse stability leads to improved performance in practice. Note that there already are results using, e.g. *cosine similarity*, as regularizer to prevent parallel weight vectors [4, 35] as well as approaches, called *Sobolev Training*, reporting better generalization and data-efficiency by employing a Sobolev norm based loss [12].

[4]This could be of interest in the design of dynamic network architectures [26, 28, 40] and is also closely related to the co-adaption of neurons, to counteract which, dropout was invented [21].

Acknowledgment

The research of JB and DE was supported by the Austrian Science Fund (FWF) under grants I3403-N32 and P 30148. The authors would like to thank Pavol Harár for helpful comments.

References

[1] Z. Allen-Zhu, Y. Li, and Z. Song. A Convergence Theory for Deep Learning via Over-Parameterization. *arXiv:1811.03962*, 2018.

[2] M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 2009.

[3] S. Arora, R. Ge, B. Neyshabur, and Y. Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*, pages 254–263, 2018.

[4] N. Bansal, X. Chen, and Z. Wang. Can we gain more from orthogonality regularizations in training deep networks? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 4261–4271. Curran Associates, Inc., 2018.

[5] P. L. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. *arXiv:1706.08498*, 2017.

[6] P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *arXiv:1703.02930*, 2017.

[7] J. Berner, P. Grohs, and A. Jentzen. Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations. *arXiv:1809.03062*, 2018.

[8] J. Berner, D. Elbrächter, P. Grohs, and A. Jentzen. Towards a regularity theory for ReLU networks–chain rule and global error estimates. *arXiv:1905.04992*, 2019.

[9] H. Bölcskei, P. Grohs, G. Kutyniok, and P. Petersen. Optimal approximation with sparsely connected deep neural networks. *arXiv:1705.01714*, 2017.

[10] M. Burger and A. Neubauer. Error Bounds for Approximation with Neural Networks . *Journal of Approximation Theory*, 112(2):235–250, 2001.

[11] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

[12] W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu. Sobolev training for neural networks. In *Advances in Neural Information Processing Systems*, pages 4278–4287, 2017.

[13] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai. Gradient Descent Finds Global Minima of Deep Neural Networks. *arXiv:1811.03804*, 2018.

[14] L. C. Evans. *Partial Differential Equations (second edition)*. Graduate studies in mathematics. American Mathematical Society, 2010.

[15] L. C. Evans and R. F. Gariepy. *Measure Theory and Fine Properties of Functions, Revised Edition*. Textbooks in Mathematics. CRC Press, 2015.

[16] K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183–192, 1989.

[17] N. Golowich, A. Rakhlin, and O. Shamir. Size-independent sample complexity of neural networks. *arXiv:1712.06541*, 2017.

[18] I. J. Goodfellow, O. Vinyals, and A. M. Saxe. Qualitatively characterizing neural network optimization problems. *arXiv:1412.6544*, 2014.

[19] R. Gribonval, G. Kutyniok, M. Nielsen, and F. Voigtlaender. Approximation spaces of deep neural networks. *arXiv: 1905.01208*, 2019.

[20] I. Gühring, G. Kutyniok, and P. Petersen. Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms. *arXiv:1902.07896*, 2019.

[21] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012.

[22] K. Kawaguchi. Deep learning without poor local minima. In *Advances in neural information processing systems*, pages 586–594, 2016.

[23] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399, 2018.

[24] Y. Li and Y. Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pages 8157–8166, 2018.

[25] Y. Li and Y. Yuan. Convergence analysis of two-layer neural networks with ReLU activation. In *Advances in Neural Information Processing Systems*, pages 597–607, 2017.

[26] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. *arXiv:1806.09055*, 2018.

[27] S. Mei, A. Montanari, and P.-M. Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

[28] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Chapter 15 - evolving deep neural networks. In R. Kozma, C. Alippi, Y. Choe, and F. C. Morabito, editors, *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pages 293 – 312. Academic Press, 2019.

[29] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017.

[30] Q. Nguyen and M. Hein. The loss surface of deep and wide neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 2603–2612. JMLR.org, 2017.

[31] J. Pennington and Y. Bahri. Geometry of neural network loss surfaces via random matrix theory. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 2798–2806. JMLR.org, 2017.

[32] D. Perekrestenko, P. Grohs, D. Elbrächter, and H. Bölcskei. The universal approximation power of finite-width deep ReLU networks. *arXiv:1806.01528*, 2018.

[33] P. Petersen and F. Voigtlaender. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *arXiv:1709.05289*, 2017.

[34] P. Petersen, M. Raslan, and F. Voigtlaender. Topological properties of the set of functions generated by neural networks of fixed size. *arXiv:1806.08459*, 2018.

[35] P. Rodríguez, J. Gonzalez, G. Cucurull, J. M. Gonfaus, and X. Roca. Regularizing cnns with locally constrained decorrelations. *arXiv:1611.01967*, 2016.

[36] I. Safran and O. Shamir. On the quality of the initial basin in overspecified neural networks. In *International Conference on Machine Learning*, pages 774–782, 2016.

[37] U. Shaham, A. Cloninger, and R. R. Coifman. Provable approximation properties for deep neural networks. *Applied and Computational Harmonic Analysis*, 44(3):537 – 557, 2018.

[38] O. Shamir and T. Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International Conference on Machine Learning*, pages 71–79, 2013.

[39] D. Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94: 103–114, 2017.

[40] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

A Appendix - Proofs and Additional Material

A.1 Section 1

A.1.1 Additional Material

Example A.1 (Without inverse stability: parameter minimum $\not\Longrightarrow$ realization minimum)**.** *Consider the two domains*

$$D_1 := \{(x_1, x_2) \in (-1,1)^2 \colon x_2 > |x_1|\}, \quad D_2 := \{(x_1, x_2) \in (-1,1)^2 \colon x_1 > |x_2|\}. \tag{40}$$

For simplicity of presentation, assume we are given two samples $x^1 \in D_1$, $x^2 \in D_2$ with labels $y^1 = 0$, $y^2 = 1$. The corresponding MSE is

$$\mathcal{L}(g) = \tfrac{1}{2}\big((g(x^1))^2 + (g(x^2) - 1)^2\big) \tag{41}$$

for every $g \in C(\mathbb{R}^2, \mathbb{R})$. Let the zero realization be parametrized by[5]

$$\Gamma_* = (0, (-1, 0)) \in \mathcal{N}_{(2,1,1)} \tag{42}$$

with loss $\mathcal{L}(\mathcal{R}(\Gamma_*)) = \frac{1}{2}$. Note that changing each weight by less than $\frac{1}{2}$ does not decrease the loss, as this rotates the vector $(-1, 0)$ by at most $45°$. Thus Γ_* is a local minimum in the parametrization space. However, the sequence of realizations given by

$$g_k(x) = \tfrac{1}{k}\rho(x_1 - x_2) = \mathcal{R}((1, -1), \tfrac{1}{k}) \tag{43}$$

satisfies that

$$\|g_k - \mathcal{R}(\Gamma_*)\|_{W^{1,\infty}((-1,1)^2)} = \|g_k\|_{W^{1,\infty}((-1,1)^2)} \leq \tfrac{1}{k} \tag{44}$$

and

$$\mathcal{L}(g_k) = \tfrac{1}{2}(g_k(x^2) - 1)^2 < \tfrac{1}{2} = \mathcal{L}(\mathcal{R}(\Gamma_*)), \tag{45}$$

see Figure 6. Accordingly, $\mathcal{R}(\Gamma_*)$ is not *a local minimum in the realization space even w.r.t. the Sobolev norm. The problem occurs, since inverse stability fails due to unbalancedness of Γ_*.*

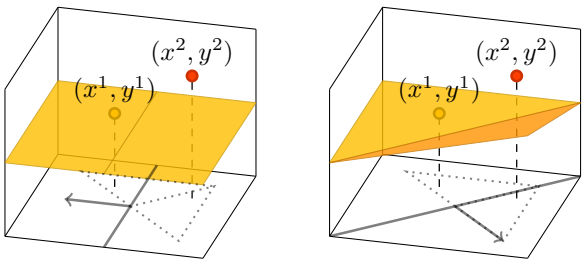


Figure 6: The figure shows the samples $((x^i, y^i))_{i=1,2}$, the realization $\mathcal{R}(\Gamma_*)$ of the local parameter minimum (left) and g_3 (right).

Theorem A.2 (Quality of local realization minima)**.** *Assume that*

$$\sup_{f \in S} \inf_{\Phi \in \Omega_N} \|\mathcal{R}(\Phi) - f\| < \eta \quad \textit{(approximability)}. \tag{46}$$

Let g_* be a local minimum with radius $r' \geq 2\eta$ of the optimization problem $\min_{g \in \mathcal{R}(\Omega_N)} \mathcal{L}(g)$. Then it holds for every $g \in \mathcal{R}(\Omega_N)$ (in particular for every global minimizer) that

$$\mathcal{L}(g_*) \leq \mathcal{L}(g) + \tfrac{2c}{r'}\|g_* - g\|\eta. \tag{47}$$

Proof. Define $\lambda := \frac{r'}{2\|g - g_*\|}$ and $f := (1 - \lambda)g_* + \lambda g \in S$. Due to (46) there is $\Phi \in \Omega_N$ such that $\|\mathcal{R}(\Phi) - f\| \leq \eta$ and by the assumptions on g_* and $\mathcal{L}$ it holds that

$$\mathcal{L}(g_*) \leq \mathcal{L}(\mathcal{R}(\Phi)) \leq \mathcal{L}(f) + c\eta \leq (1 - \lambda)\mathcal{L}(g_*) + \lambda\mathcal{L}(g) + c\eta.$$

This completes the proof. See Figure 7 for illustration. □

[5] See notation in the beginning of Section 2.

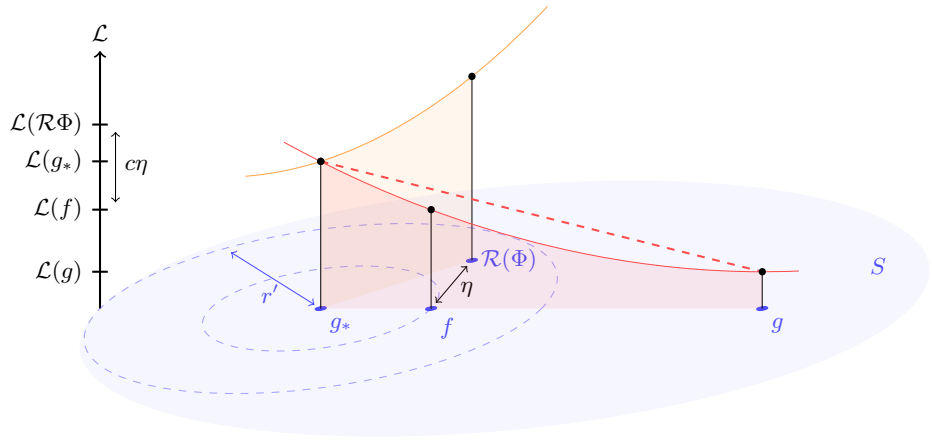


Figure 7: The figure illustrates the proof idea of Theorem A.2. Note that decreasing η, c, $\|g_* - g\|$ or increasing r' leads to a better local minimum due to the convexity of the loss function (red).

A.1.2 Proofs

Proof of Proposition 1.2. By Definition 1.1 we know that for every $g \in \mathcal{R}(\Omega)$ with $\|g - \mathcal{R}(\Gamma_*)\| \leq (\frac{r}{s})^{1/\alpha}$ there exists $\Phi \in \Omega$ with

$$\mathcal{R}(\Phi) = g \quad \text{and} \quad \|\Phi - \Gamma_*\|_\infty \leq s\|g - \mathcal{R}(\Gamma_*)\|^\alpha \leq r. \tag{48}$$

Therefore by assumption it holds that

$$\mathcal{L}(\mathcal{R}(\Gamma_*)) \leq \mathcal{L}(\mathcal{R}(\Phi)) = \mathcal{L}(g). \tag{49}$$

which proves the claim. □

Proof of Theorem 1.3. Let $\varepsilon, r > 0$, define $r' := (\frac{r}{s})^{1/\alpha}$ and $\eta := \min\{(\frac{2c}{r'}\operatorname{diam}(S))^{-1}\varepsilon, \frac{r'}{2}\}$. Then compactness of S implies the existence of an architecture $n(\varepsilon, r) \in \mathcal{A}_L$ such that for every $N \in \mathcal{A}_L$ with $N_1 \geq n_1(\varepsilon, r), \ldots, N_{L-1} \geq n_{L-1}(\varepsilon, r)$ the approximability assumption (46) is satisfied. Let now Γ_* be a local minimum with radius at least r of $\min_{\Gamma \in \Omega_N} \mathcal{L}(\mathcal{R}(\Gamma))$. As we assume uniform (s, α) inverse stability, Proposition 1.2 implies that $\mathcal{R}(\Gamma_*)$ is a local minimum of the optimization problem $\min_{g \in \mathcal{R}(\Omega_N)} \mathcal{L}(g)$ with radius at least $r' = (\frac{r}{s})^{1/\alpha} \geq 2\eta$. Theorem A.2 establishes the claim. □

Proof of Corollary 1.4. We simply combine the main observations from our paper. First, note that the assumptions imply that the restricted parametrization space Ω, which we are optimizing over, is the space $\mathcal{N}^*_{(d+2,N_1+1,D)}$ from Definition 3.2. Secondly, Theorem 3.3 implies that the realization map is $(4, 1/2)$ inverse stable on Ω. Thus, Proposition 1.2 directly proves Claim 1. For the proof of Claim 2 we make use of Lemma A.6. It implies that for every $\Theta \in \mathcal{P}_{(d,N_1,D)}$ there exists $\Gamma \in \Omega$ such that it holds that

$$\tfrac{1}{n}\sum_{i=1}^{n} \|\mathcal{R}(\Gamma)(\tilde{x}^i) - y^i\|^2 = \tfrac{1}{n}\sum_{i=1}^{n} \|\mathcal{R}(\Theta)(x^i) - y^i\|^2, \tag{50}$$

which proves the claim. □

A.2 Section 2

A.2.1 Additional Material

Lemma A.3 (Reparametrization in case of linearly independent weight vectors)**.** *Let*

$$\Theta = (A^\Theta, C^\Theta) = \left([a_1^\Theta| \ldots |a_m^\Theta]^T, [c_1^\Theta| \ldots |c_m^\Theta]\right) \in \mathcal{N}_{(d,m,D)} \tag{51}$$

with linearly independent weight vectors $(a_i^\Theta)_{i=1}^m$ *and* $\min_{i\in[m]} \|c_i^\Theta\|_\infty > 0$ *and let*

$$\Phi = (A^\Phi, B^\Phi) = \big([a_1^\Phi|\dots|a_m^\Phi]^T, [c_1^\Phi|\dots|c_m^\Phi]\big) \in \mathcal{N}_{(d,m,D)} \tag{52}$$

with $\mathcal{R}(\Phi) = \mathcal{R}(\Theta)$. *Then there exists a permutation* $\pi\colon [m] \to [m]$ *such that for every* $i \in [m]$ *there exist* $\lambda_i \in (0,\infty)$ *with*

$$a_i^\Phi = \lambda_i a_{\pi(i)}^\Theta \quad \textit{and} \quad c_i^\Phi = \tfrac{1}{\lambda_i} c_{\pi(i)}^\Theta. \tag{53}$$

This means that, up to reordering and rebalancing, Θ *is the unique parametrization of* $\mathcal{R}(\Theta)$.

Proof. First we define for every $s \in \{0,1\}^m$ the corresponding open orthant

$$O^s := \{x \in \mathbb{R}^m\colon x_1(2s_1-1) > 0, \dots, x_m(2s_m-1) > 0\} \subseteq \mathbb{R}^m. \tag{54}$$

By assumption A^Θ has rank m, i.e. is surjective, and therefore the preimages of the orthants

$$H^s := \{x \in \mathbb{R}^d\colon A^\Theta x \in O^s\} \subseteq \mathbb{R}^d, \quad s \in \{0,1\}^m, \tag{55}$$

are disjoint, non-empty open sets. Note that on each H^s the realization $\mathcal{R}(\Theta)$ is linear with

$$\mathcal{R}(\Theta)(x) = C^\Theta \operatorname{diag}(s) A^\Theta x \quad \text{and} \quad D\mathcal{R}(\Theta)(x) = C^\Theta \operatorname{diag}(s) A^\Theta. \tag{56}$$

Since A^Θ has full row rank, it has a right inverse. Thus we have for $s,t \in \{0,1\}^m$ that

$$C^\Theta \operatorname{diag}(s) A^\Theta = C^\Theta \operatorname{diag}(t) A^\Theta \implies C^\Theta \operatorname{diag}(s) = C^\Theta \operatorname{diag}(t). \tag{57}$$

Note that $C^\Theta \operatorname{diag}(s) = C^\Theta \operatorname{diag}(t)$ can only hold if $s = t$ due to the assumptions that $\|c_i^\Theta\|_\infty \neq 0$ for all $i \in [m]$. Thus the above establishes that for $s,t \in \{0,1\}^m$ it holds that

$$C^\Theta \operatorname{diag}(s) A^\Theta = C^\Theta \operatorname{diag}(t) A^\Theta \quad \text{if and only if} \quad s = t, \tag{58}$$

i.e. $\mathcal{R}(\Theta)$ has different derivatives on its 2^m linear regions. In order for $\mathcal{R}(\Phi)$ to have matching linear regions and matching derivatives on each one of them, there must exist a permutation matrix $P \in \{0,1\}^{m\times m}$ such that for every $s \in \{0,1\}^m$

$$PA^\Phi x \in O^s \quad \text{for every } x \in H^s. \tag{59}$$

Thus, there exist $(\lambda_i)_{i=1}^m \in (0,\infty)^m$ such that

$$A^\Phi = \operatorname{diag}(\lambda_1, \dots, \lambda_m) P^T A^\Theta. \tag{60}$$

The assumption that $D\mathcal{R}(\Theta) = D\mathcal{R}(\Psi)$, together with (56) for $s = (1,\dots,1)$, implies that

$$C^\Phi = C^\Theta P \operatorname{diag}(\tfrac{1}{\lambda_1}, \dots, \tfrac{1}{\lambda_m}), \tag{61}$$

which proves the claim. □

Example A.4 (Failure due to unbalancedness)**.** *Let*

$$\Gamma_k := \big((k,0), \tfrac{1}{k^2}\big) \in \mathcal{N}_{(2,1,1)}, \quad k \in \mathbb{N}, \tag{62}$$

and $g_k \in \mathcal{R}(\mathcal{N}_{(2,1,1)})$ *be given by*

$$g_k(x) = \tfrac{1}{k}\rho(\langle (0,1), x\rangle), \quad k \in \mathbb{N}. \tag{63}$$

The only way to parametrize g_k *is* $g_k(x) = \mathcal{R}(\Phi_k)(x) = c\rho(\langle (0,a), x\rangle)$ *with* $a, c > 0$ *(see Lemma A.3), and we have*

$$|\mathcal{R}(\Phi_k) - \mathcal{R}(\Gamma_k)|_{W^{1,\infty}} \leq \tfrac{1}{k} \quad \textit{and} \quad \|\Phi_k - \Gamma_k\|_\infty \geq k. \tag{64}$$

Lemma A.5. *Let* $d, m \in \mathbb{N}$ *and* $a_i \in \mathbb{R}^d$, $i \in [m]$, *such that* $\sum_{i\in[m]} a_i = 0$. *Then it holds for all* $x \in \mathbb{R}^d$ *that*

$$\sum_{i\in[m]} \rho(\langle a_i, x\rangle) = \sum_{i\in[m]} \rho(\langle -a_i, x\rangle). \tag{65}$$

Proof. By assumption we have for all $x \in \mathbb{R}^d$ that $\sum_{i\in[m]} \langle a_i, x\rangle = 0$. This implies for all $x \in \mathbb{R}^d$ that

$$\sum_{i\in[m]\colon \langle a_i,x\rangle \geq 0} \langle a_i, x\rangle - \sum_{i\in[m]} \langle a_i, x\rangle = \sum_{i\in[m]\colon \langle a_i,x\rangle \leq 0} -\langle a_i, x\rangle, \tag{66}$$

which proves the claim. □

A.2.2 Proofs

Proof of Example 2.1. We have for every $k \in \mathbb{N}$ that

$$\|g_k\|_{L^\infty((-1,1)^2)} \leq \tfrac{1}{k} \quad \text{and} \quad |g_k|_{W^{1,\infty}} = k^2. \tag{67}$$

Assume that there exists sequence of networks $(\Phi_k)_{k\in\mathbb{N}} \subseteq \mathcal{N}_{(2,2,1)}$ with $\mathcal{R}(\Phi_k) = g_k$ and with uniformly bounded parameters, i.e. $\sup_{k\in\mathbb{N}} \|\Phi_k\|_\infty < \infty$. Note that there exists a constant C (depending only on the network architecture) such that the realizations $\mathcal{R}(\Phi_k)$ are Lipschitz continuous with

$$\mathrm{Lip}(\mathcal{R}(\Phi_k)) \leq C\|\Phi_k\|_\infty^2$$

(see [34, Prop. 5.1]). It follows that $|\mathcal{R}(\Phi_k)|_{W^{1,\infty}} \leq \mathrm{Lip}(\mathcal{R}(\Phi_k))$ is uniformly bounded which contradicts (67). □

Proof of Example 2.2. The only way to parametrize g_k is $g_k(x) = \mathcal{R}(\Phi_k)(x) = c\rho(\langle(0,a),x\rangle)$ with $a, c > 0$ (see also Lemma A.3), which proves the claim. □

Proof of Example 2.3. Any parametrization of g_k must be of the form $\Phi_k := (A, c) \in \mathbb{R}^{2\times 2} \times \mathbb{R}^{1\times 2}$ with

$$A = \begin{bmatrix} a_1 & 0 \\ 0 & a_2 \end{bmatrix} \quad \text{or} \quad A = \begin{bmatrix} 0 & a_2 \\ a_1 & 0 \end{bmatrix} \tag{68}$$

(see Lemma A.3). Thus it holds that $\|\Phi_k - \Gamma\|_\infty \geq \|(1,0) - (0,a_2)\|_\infty \geq 1$ and the proof is completed by direct calculation. □

Proof of Example 2.4. Let Φ_k be an arbitrary parametrization of g_k given by

$$\Phi_k = \left([\tilde{a}_1|\tilde{a}_2|\dots|\tilde{a}_{2m}]^T, \tilde{c}\right) \in \mathcal{N}_{(d,2m,1)} \tag{69}$$

As g_k has two linear regions separated by the hyperplane with normal vector v, there exists $j \in [2m]$ and $\lambda \in \mathbb{R} \setminus \{0\}$ such that

$$\tilde{a}_j = \lambda v. \tag{70}$$

The distance of any weight vector $\pm a_i$ of Γ to the line $\{\lambda v \colon \lambda \in \mathbb{R}\}$ can be lower bounded by

$$\| \pm a_i - \lambda v\|_\infty^2 \geq \tfrac{1}{d}\| \pm a_i - \lambda v\|_2^2 \geq \tfrac{1}{d^2}\left[\|a_i\|_2^2\|v\|_2^2 - \langle a_i, v\rangle^2\right], \quad i \in [m], \lambda \in \mathbb{R}. \tag{71}$$

The Cauchy-Schwarz inequality and the linear independence of v to each a_i, $i \in [m]$, establishes that $C := \frac{1}{d^2}\min_{i\in[m]}\left[\|a_i\|_2^2\|v\|_2^2 - \langle a_i, v\rangle^2\right] > 0$. Together with the fact that $\mathcal{R}(\Gamma) = 0$, this completes the proof. □

Proof of Example 2.5. Since $x = \rho(x) - \rho(-x)$ for every $x \in \mathbb{R}$, the difference of the realizations is linear, i.e.

$$\mathcal{R}(\Theta_k) - \mathcal{R}(\Gamma_k) = \langle c_1^k a_1^k + c_2^k a_2^k + c_3^k a_3^k, x\rangle = \langle(0,0,3),x\rangle \tag{72}$$

and thus the difference of the gradients is constant, i.e.

$$|\mathcal{R}(\Theta_k) - \mathcal{R}(\Gamma_k)|_{W^{1,\infty}} = 3, \quad k \in \mathbb{N}. \tag{73}$$

However, regardless of the balancing and reordering of the weight vectors a_i^k, $i \in [3]$, we have that

$$\|\Theta_k - \Gamma_k\|_\infty \geq k. \tag{74}$$

By Lemma A.3, up to balancing and reordering, there does not exist any other parametrization of Θ_k with the same realization. □

A.3 Section 3

A.3.1 Additional Material

Lemma A.6. *Let $d, m, D \in \mathbb{N}$ and $\Theta \in \mathcal{P}_{(d,m,D)}$. Then there exists $\Gamma \in \mathcal{N}^*_{(d+2,m+1,D)}$ such that for all $x \in \mathbb{R}^d$ it holds that*

$$\mathcal{R}(\Gamma)(x_1, \dots, x_d, 1, -1) = \mathcal{R}(\Theta)(x). \tag{75}$$

Proof. Since $\Theta \in \mathcal{P}_{(d,m,D)}$ it can be written as

$$\Theta = \Big(\big(A, b\big), \big(c, e\big)\Big) = \Big(\big([a_1|\dots|a_m]^T, b\big), \big([c_1|\dots|c_m], e\big)\Big) \tag{76}$$

with

$$\mathcal{R}(\Theta)(x) = \sum_{i=1}^{m} c_i \rho(\langle a_i, x\rangle + b_i) + e, \quad x \in \mathbb{R}^d, \tag{77}$$

where $A \in \mathbb{R}^{m\times d}$, $b \in \mathbb{R}^m$, $C \in \mathbb{R}^{D\times m}$, and $e \in \mathbb{R}^D$. We define for $i \in [m]$

$$b_i^+ := \begin{cases} b_i + 1 & : b_i \geq 0 \\ 1 & : b_i < 0 \end{cases}, \quad \text{and} \quad b_i^- := \begin{cases} 1 & : b_i \geq 0 \\ -b_i + 1 & : b_i < 0 \end{cases} \tag{78}$$

and observe that $b_i^+ > 0$, $b_i^- > 0$, and $b_i^+ - b_i^- = b_i$. For $i \in [m]$ let

$$c_i^* := \begin{cases} c_i & : \|c_i\|_\infty \neq 0 \\ (1, \dots, 1) & : \|c_i\|_\infty = 0 \end{cases} \tag{79}$$

and

$$a_i^* := \begin{cases} (a_{i,1}, \dots, a_{i,d}, b_i^+, b_i^-) & : \|c_i\|_\infty \neq 0 \\ (0, \dots, 0, 1, 1) & : \|c_i\|_\infty = 0 \end{cases}. \tag{80}$$

Note that we have

$$\mathcal{R}(\Theta)(x) = \sum_{i=1}^{m} c_i^* \rho(\langle a_i^*, (x_1, \dots, x_d, 1, -1)\rangle) + e, \quad x \in \mathbb{R}^d. \tag{81}$$

To include the second bias e let

$$c_{m+1}^* := \begin{cases} e & : e \neq 0 \\ (1, \dots, 1) & : e = 0 \end{cases}, \quad \text{and} \quad a_{m+1}^* := \begin{cases} (0, \dots, 0, 2, 1) & : e \neq 0 \\ (0, \dots, 0, 1, 1) & : e = 0 \end{cases}. \tag{82}$$

In order to balance the network, let $a_i^\Gamma = a_i^* (\frac{\|c_i^*\|_\infty}{\|a_i^*\|_\infty})^{1/2}$ and $c_i^\Gamma = c_i^* (\frac{\|a_i^*\|_\infty}{\|c_i^*\|_\infty})^{1/2}$ for every $i \in [m+1]$. Then the claim follows by direct computation. □

A.3.2 Proofs

Proof of Theorem 3.1. Without loss of generality[6], we can assume for all $i \in [m]$ that $a_i^\Theta = 0$ if and only if $c_i^\Theta = 0$. We now need to show that there always exists a way to reparametrize $\mathcal{R}(\Theta)$ such that the architecture remains the same and (35) is satisfied. For simplicity of notation we will write $r := |g - \mathcal{R}(\Gamma)|_{W^{1,\infty}}$ throughout the proof. Let $f_i^\Gamma \colon \mathbb{R}^d \to \mathbb{R}$ resp. $f_i^\Theta \colon \mathbb{R}^d \to \mathbb{R}$ be the part that is contributed by the i-th neuron, i.e.

$$\mathcal{R}(\Gamma) = \sum_{i=1}^{m} f_i^\Gamma \quad \text{with} \quad f_i^\Gamma(x) := c_i^\Gamma \rho(\langle a_i^\Gamma, x\rangle), \tag{83}$$

$$g = \mathcal{R}(\Theta) = \sum_{i=1}^{m} f_i^\Theta \quad \text{with} \quad f_i^\Theta(x) := c_i^\Theta \rho(\langle a_i^\Theta, x\rangle). \tag{84}$$

[6] In case one of them is zero, the other one can be set to zero without changing the realization.

Further let

$$\begin{aligned} H^+_{\Gamma,i} &:= \{x \in \mathbb{R}^d \colon \langle a^\Gamma_i, x\rangle > 0\},\\ H^0_{\Gamma,i} &:= \{x \in \mathbb{R}^d \colon \langle a^\Gamma_i, x\rangle = 0\},\\ H^-_{\Gamma,i} &:= \{x \in \mathbb{R}^d \colon \langle a^\Gamma_i, x\rangle < 0\}. \end{aligned} \tag{85}$$

By conditions C.2 and C.3a we have for all $i, j \in I^\Gamma$ that

$$i \neq j \implies H^0_{\Gamma,i} \neq H^0_{\Gamma,j}. \tag{86}$$

Further note that we can reparametrize $\mathcal{R}(\Theta)$ such that the same holds there. To this end observe that

$$c\rho(\langle a, x\rangle) + c'\rho(\langle a', x\rangle) = (c + c'\tfrac{\|a'\|_\infty}{\|a\|_\infty})\rho(\langle a, x\rangle), \tag{87}$$

given that a' is a positive multiple of a. Specifically, let $(J_k)_{k=1}^K$ be a partition of I^Θ (i.e. $J_k \neq \emptyset$, $\cup_{k=1}^K J_k = I^\Theta$ and $J_k \cap J_{k'} = \emptyset$ if $k \neq k'$), such that for all $k \in [K]$ it holds that

$$i, j \in J_k \implies \frac{a^\Theta_j}{\|a^\Theta_j\|_\infty} = \frac{a^\Theta_i}{\|a^\Theta_i\|_\infty}. \tag{88}$$

We denote by j_k the smallest element in J_k and make the following replacements, for all $i \in I^\Theta$, without changing the realization of Θ:

$$a^\Theta_i \mapsto a^\Theta_i, c^\Theta_i \mapsto \sum_{j \in J_k} c^\Theta_j \tfrac{\|a^\Theta_j\|_\infty}{\|a^\Theta_{j_k}\|_\infty}, \quad \text{if } i \in J_k \text{ and } i = j_k, \tag{89}$$

$$a^\Theta_i \mapsto 0, c^\Theta_i \mapsto 0, \qquad \text{if } i \in J_k \text{ and } i \neq j_k. \tag{90}$$

Note that we also update the set $I^\Theta := \{i \in [m] \colon a^\Theta_i \neq 0\}$ accordingly. Let now

$$\begin{aligned} H^+_{\Theta,i} &:= \{x \in \mathbb{R}^d \colon \langle a^\Theta_i, x\rangle > 0\},\\ H^0_{\Theta,i} &:= \{x \in \mathbb{R}^d \colon \langle a^\Theta_i, x\rangle = 0\},\\ H^-_{\Theta,i} &:= \{x \in \mathbb{R}^d \colon \langle a^\Theta_i, x\rangle > 0\}. \end{aligned} \tag{91}$$

By construction and condition C.3a, we have for all $i, j \in I^\Theta$ that

$$i \neq j \implies H^0_{\Theta,i} \neq H^0_{\Theta,j}. \tag{92}$$

Note that we now have a parametrization Θ of g, where all weight vectors a^Θ_i are either zero (in which case the corresponding c^Θ_i are also zero) or pairwise linearly independent to each other nonzero weight vector.
Next, for $s \in \{0,1\}^m$, let

$$\begin{aligned} H^s_\Gamma &:= \bigcap_{i \in [m] \colon s_i = 1} H^+_{\Gamma,i} \cap \bigcap_{i \in [m] \colon s_i = 0} H^-_{\Gamma,i},\\ H^s_\Theta &:= \bigcap_{i \in [m] \colon s_i = 1} H^+_{\Theta,i} \cap \bigcap_{i \in [m] \colon s_i = 0} H^-_{\Theta,i}, \end{aligned} \tag{93}$$

and

$$S^\Gamma := \{s \in \{0,1\}^m \colon H^s_\Gamma \neq \emptyset\}, \quad S^\Theta := \{s \in \{0,1\}^m \colon H^s_\Theta \neq \emptyset\}. \tag{94}$$

The H^s_Γ, $s \in S^\Gamma$, and H^s_Θ, $s \in S^\Theta$, are the interiors of the different linear regions of $\mathcal{R}(\Gamma)$ and $\mathcal{R}(\Theta)$ respectively. Next observe that the derivatives of f^Γ_i, f^Θ_i are (a.e.) given by

$$Df^\Gamma_i(x) = \mathbf{1}_{H^+_{\Gamma,i}}(x)\, c^\Gamma_i a^\Gamma_i, \quad Df^\Theta_i(x) = \mathbf{1}_{H^+_{\Theta,i}}(x)\, c^\Theta_i a^\Theta_i. \tag{95}$$

Note that for every $x \in H^s_\Gamma$, $y \in H^s_\Theta$ we have

$$\begin{aligned} D\mathcal{R}(\Gamma)(x) &= \sum_{i \in [m]} Df^\Gamma_i(x) = \sum_{i \in [m]} s_i c^\Gamma_i a^\Gamma_i =: \Sigma^\Gamma_s,\\ D\mathcal{R}(\Theta)(y) &= \sum_{i \in [m]} Df^\Theta_i(y) = \sum_{i \in [m]} s_i c^\Theta_i a^\Theta_i =: \Sigma^\Theta_s. \end{aligned} \tag{96}$$

Next we use that for $s \in S^\Gamma$, $t \in S^\Theta$ we have $|\Sigma_s^\Gamma - \Sigma_t^\Theta| \leq r$ if $H_s^\Gamma \cap H_t^\Theta \neq \emptyset$, and compare adjacent linear regions of $\mathcal{R}(\Gamma) - \mathcal{R}(\Theta)$. Let now $i \in I^\Gamma$ and consider the following cases:
Case 1: We have $H_{\Gamma,i}^0 \neq H_{\Theta,j}^0$ for all $j \in I^\Theta$. This means that the Df_k^Θ, $k \in [m]$, and the Df_k^Γ, $k \in [m]\backslash\{i\}$, are the same on both sides near the hyperplane $H_{\Gamma,i}^0$, while the value of Df_i^Γ is 0 on one side and $c_i^\Gamma a_i^\Gamma$ on the other. Specifically, there exist $s^+, s^- \in S^\Gamma$ and $s^* \in S^\Theta$ such that $s_i^+ = 1$, $s_i^- = 0$, $s_j^+ = s_j^-$ for all $j \in [m]\backslash\{i\}$, and $H_\Gamma^{s^+} \cap H_\Theta^{s^*} \neq \emptyset$, $H_\Gamma^{s^-} \cap H_\Theta^{s^*} \neq \emptyset$, which implies

$$\|c_i^\Gamma a_i^\Gamma\|_\infty = \|(\Sigma_{s^+}^\Gamma - \Sigma_{s^*}^\Theta) - (\Sigma_{s^-}^\Gamma - \Sigma_{s^*}^\Theta)\|_\infty \leq 2r. \tag{97}$$

Case 2: There exists $j \in I^\Theta$ such that $H_{\Gamma,i}^0 = H_{\Theta,j}^0$. Note that (86) ensures that $H_{\Gamma,i}^0 \neq H_{\Gamma,k}^0$ for $k \in [m] \setminus \{i\}$ and (92) ensures that $H_{\Theta,j}^0 \neq H_{\Gamma,k}^0$ for $k \in [m] \setminus \{j\}$. Moreover, Condition C.3b implies $H_{\Gamma,i}^+ = H_{\Theta,j}^+$. This means that the Df_k^Θ, $k \in [m]\backslash\{j\}$, and the Df_k^Γ, $k \in [m]\backslash\{i\}$, are the same on both sides near the hyperplane $H_{\Gamma,i}^0 = H_{\Theta,j}^0$, while the values of Df_i^Γ and Df_j^Θ change. Specifically there exist $s^+, s^- \in S^\Gamma$ and $t^+, t^- \in S^\Theta$ such that $s_i^+ = 1$, $s_i^- = 0$, $s_k^+ = s_k^-$ for all $k \in [m]\backslash\{i\}$, $t_j^+ = 1$, $t_j^- = 0$, $t_k^+ = t_k^-$ for all $k \in [m]\backslash\{j\}$ and $H_{s^+}^\Gamma \cap H_{t^+}^\Theta \neq \emptyset$, $H_{s^-}^\Gamma \cap H_{t^-}^\Theta \neq \emptyset$, which implies

$$\|c_i^\Gamma a_i^\Gamma - c_j^\Theta a_j^\Theta\|_\infty = \|(\Sigma_{s^+}^\Gamma - \Sigma_{t^+}^\Theta) - (\Sigma_{s^-}^\Gamma - \Sigma_{t^-}^\Theta)\|_\infty \leq 2r. \tag{98}$$

Analogously we get for $i \in I^\Theta$ that $H_{\Theta,i}^0 \neq H_{\Gamma,j}^0$ for all $j \in I^\Gamma$ implies $\|c_i^\Theta a_i^\Theta\|_\infty \leq 2r$. Next let

$$I_1 := \{i \in [m] \colon H_{\Gamma,i}^0 \neq H_{\Theta,j}^0 \text{ for all } j \in I^\Theta\} \cup \{i \in [m] \colon a_i^\Gamma = 0\} \tag{99}$$

and

$$I_2 := [m] \setminus I_1 = \{i \in [m] \colon \exists\, j \in I^\Theta \text{ such that } H_{\Gamma,i}^+ = H_{\Theta,j}^+\}. \tag{100}$$

Colloquially speaking, this shows that for every f_i^Γ with $i \in I_2$ there is a f_j^Θ with exactly matching half-spaces, i.e. $H_{\Gamma,i}^+ = H_{\Theta,j}^+$, and approximately matching gradients (Case 2). Moreover, all unmatched f_i^Γ and f_j^Θ must have a small gradient (Case 1).
Specifically, the above establishes that there exists a permutation $\pi \colon [m] \to [m]$ such that for every $i \in I_1$ it holds that

$$\|c_i^\Gamma a_i^\Gamma\|_\infty, \|c_{\pi(i)}^\Theta a_{\pi(i)}^\Theta\|_\infty \leq 2r, \tag{101}$$

and for every $i \in I_2$ that

$$\|c_i^\Gamma a_i^\Gamma - c_{\pi(i)}^\Theta a_{\pi(i)}^\Theta\|_\infty \leq 2r. \tag{102}$$

We make the following replacements, for all $i \in [m]$, without changing the realization of Θ:

$$a_i^\Theta \to a_{\pi(i)}^\Theta, \quad c_i^\Theta \to c_{\pi(i)}^\Theta. \tag{103}$$

In order to balance the weights of Θ for I_1, we further make the following replacements, for all $i \in I_1$ with $a_i^\Theta \neq 0$, without changing the realization of Θ:

$$a_i^\Theta \to \left(\tfrac{|c_i^\Theta|}{\|a_i^\Theta\|_\infty}\right)^{1/2} a_i^\Theta, \quad c_i^\Theta \to \left(\tfrac{\|a_i^\Theta\|_\infty}{|c_i^\Theta|}\right)^{1/2} c_i^\Theta. \tag{104}$$

This implies for every $i \in I_1$ that

$$|c_i^\Theta|, \|a_i^\Theta\|_\infty \leq (2r)^{1/2}. \tag{105}$$

Moreover, due to Condition C.1, we get for every $i \in I_1$ that

$$|c_i^\Gamma|, \|a_i^\Gamma\|_\infty \leq \beta. \tag{106}$$

Thus we get for every $i \in I_1$ that

$$|c_i^\Theta - c_i^\Gamma|, \|a_i^\Theta - a_i^\Gamma\|_\infty \leq \beta + (2r)^{1/2}. \tag{107}$$

Next we (approximately) match the balancing of (c_i^Θ, a_i^Θ) to the balancing of (c_i^Γ, a_i^Γ) for $i \in I_2$, in order to derive estimates on $|c_i^\Theta - c_i^\Gamma|$ and $\|a_i^\Theta - a_i^\Gamma\|_\infty$ from (102). Specifically, we make the following replacements, for all $i \in I_2$, without changing the realization of Θ:

$$a_i^\Theta \to (\tfrac{|c_i^\Theta|}{\|a_i^\Theta\|_\infty})^{1/2}\, a_i^\Theta, \quad c_i^\Theta \to (\tfrac{\|a_i^\Theta\|_\infty}{|c_i^\Theta|})^{1/2}\, c_i^\Theta, \qquad \text{if } \|c_i^\Gamma a_i^\Gamma\|_\infty \le 2r, \tag{108}$$

$$a_i^\Theta \to \frac{c_i^\Theta}{c_i^\Gamma} a_i^\Theta, \quad c_i^\Theta \to c_i^\Gamma, \qquad \text{if } \|c_i^\Gamma a_i^\Gamma\|_\infty > 2r, |c_i^\Gamma| > \|a_i^\Gamma\|_\infty, \tag{109}$$

$$a_i^\Theta \to a_i^\Gamma, \quad c_i^\Theta \to \frac{\|a_i^\Theta\|_\infty}{\|a_i^\Gamma\|_\infty} c_i^\Theta, \qquad \text{if } \|c_i^\Gamma a_i^\Gamma\|_\infty > 2r, |c_i^\Gamma| < \|a_i^\Gamma\|_\infty, \tag{110}$$

$$a_i^\Theta \to (\tfrac{|c_i^\Theta|}{\|a_i^\Theta\|_\infty})^{1/2}\, a_i^\Theta, \quad c_i^\Theta \to (\tfrac{\|a_i^\Theta\|_\infty}{|c_i^\Theta|})^{1/2}\, c_i^\Theta, \quad \text{if } \|c_i^\Gamma a_i^\Gamma\|_\infty > 2r, |c_i^\Gamma| = \|a_i^\Gamma\|_\infty. \tag{111}$$

Let now $i \in I_2$ and consider the following cases:
Case A: We have $\|c_i^\Gamma a_i^\Gamma\|_\infty \le 2r$ which, together with (102), implies $\|c_i^\Theta a_i^\Theta\|_\infty \le 4r$. Due to (108) and Condition C.1 it follows that

$$|c_i^\Theta - c_i^\Gamma|, \|a_i^\Theta - a_i^\Gamma\|_\infty \le \beta + 2r^{1/2}. \tag{112}$$

Case B.1: We have $\|c_i^\Gamma a_i^\Gamma\|_\infty > 2r$ and $|c_i^\Gamma| > \|a_i^\Gamma\|_\infty$ which ensures $|c_i^\Gamma| > \|c_i^\Gamma a_i^\Gamma\|_\infty^{1/2}$. Due to (109) we get $c_i^\Theta = c_i^\Gamma$ and it follows that

$$\|a_i^\Theta - a_i^\Gamma\|_\infty = \frac{1}{|c_i^\Gamma|}\|c_i^\Theta a_i^\Theta - c_i^\Gamma a_i^\Gamma\|_\infty \le \frac{2r}{\|c_i^\Gamma a_i^\Gamma\|_\infty^{1/2}} \le (2r)^{1/2}. \tag{113}$$

Case B.2: We have $\|c_i^\Gamma a_i^\Gamma\|_\infty > 2r$ and $|c_i^\Gamma| < \|a_i^\Gamma\|_\infty$ which ensures $\|a_i^\Gamma\| > \|c_i^\Gamma a_i^\Gamma\|_\infty^{1/2}$. Due to (110) we get $a_i^\Theta = a_i^\Gamma$ and it follows that

$$|c_i^\Theta - c_i^\Gamma| = \frac{1}{\|a_i^\Gamma\|_\infty}\|c_i^\Theta a_i^\Theta - c_i^\Gamma a_i^\Gamma\|_\infty \le \frac{2r}{\|c_i^\Gamma a_i^\Gamma\|_\infty^{1/2}} \le (2r)^{1/2}. \tag{114}$$

Case B.3: We have $\|c_i^\Gamma a_i^\Gamma\|_\infty > 2r$ and $|c_i^\Gamma| = \|a_i^\Gamma\|_\infty$. Note that $\|c_i^\Gamma a_i^\Gamma\|_\infty > 2r$ and (102) ensure that $\operatorname{sgn}(c_i^\Theta) = \operatorname{sgn}(c_i^\Gamma)$, and that for $x, y > 0$ it holds that $|x - y| \le |x^2 - y^2|^{1/2}$. Combining this with the definition of I_2, the reverse triangle inequality, and (111) implies that

$$\|a_i^\Theta - a_i^\Gamma\|_\infty \le (2r)^{1/2} \quad \text{and} \quad \left|c_i^\Theta - c_i^\Gamma\right| \le (2r)^{1/2}. \tag{115}$$

Combining (107), (112), (113), (114), and (115) establishes that

$$\|\Theta - \Gamma\|_\infty \le \beta + 2r^{\frac{1}{2}}, \tag{116}$$

which completes the proof. □

Proof of Theorem 3.3. Let $\Theta \in \mathcal{N}_N^*$ be a parametrization of g, i.e. $\mathcal{R}(\Theta) = g$. We write

$$\Gamma = \left(\left[\begin{array}{c} a_1^\Gamma \\ \hline \vdots \\ \hline a_m^\Gamma \end{array}\right], \left[c_1^\Gamma \middle| \dots \middle| c_m^\Gamma\right] \right), \quad \Theta = \left(\left[\begin{array}{c} a_1^\Theta \\ \hline \vdots \\ \hline a_m^\Theta \end{array}\right], \left[c_1^\Theta \middle| \dots \middle| c_m^\Theta\right] \right) \in \mathcal{N}_{(d,m,D)}^* \tag{117}$$

and $r := |g - \mathcal{R}(\Gamma)|_{W^{1,\infty}}$. For convenience of notation we consider the weight vectors a_i^Γ, a_i^Θ here as row vectors in order to write the derivatives of the ridge functions as $c_i^\Gamma a_i^\Gamma,\ c_i^\Theta a_i^\Theta \in \mathbb{R}^{D\times d}$ without transposing.
We will now adjust the approach used in the proof of Theorem 3.1 to work for multi-dimensional outputs in the case of balanced networks. By definition of $\mathcal{N}_N^*$, the $(a_i^\Theta)_{i=1}^m$ are pairwise linearly independent and we can skip the first reparametrization step in (89) and (90).
The following "hyperplane-jumping" argument, which was used to get the estimates (97) and (98), works analogously since Conditions C.2 and C.3 are fulfilled by definition of $\mathcal{N}_N^*$. This establishes the existence of a permutation $\pi\colon [m] \to [m]$ and sets $I_1, I_2 \subseteq [m]$, as defined as in (99) and (100), such that for every $i \in I_1$ it holds that

$$\|c_i^\Gamma a_i^\Gamma\|_\infty, \|c_{\pi(i)}^\Theta a_{\pi(i)}^\Theta\|_\infty \le 2r, \tag{118}$$

and for every $i \in I_2$ that

$$\|c_i^\Gamma a_i^\Gamma - c_{\pi(i)}^\Theta a_{\pi(i)}^\Theta\|_\infty \leq 2r. \tag{119}$$

As in (103), we make the following replacements, for all $i \in [m]$, without changing the realization of Θ:

$$a_i^\Theta \to a_{\pi(i)}^\Theta, \quad c_i^\Theta \to c_{\pi(i)}^\Theta. \tag{120}$$

Note that the weights of Θ are already balanced, i.e. we have for every $i \in [m]$ that

$$\|c_i^\Theta\|_\infty = \|a_i^\Theta\|_\infty = \|c_i^\Theta\|_\infty^{1/2}\|a_i^\Theta\|_\infty^{1/2} = \|c_i^\Theta a_i^\Theta\|_\infty^{1/2}. \tag{121}$$

Thus, we can skip the reparametrization step in (104) and get directly for every $i \in I_1$ that

$$\|c_i^\Theta - c_i^\Gamma\|_\infty \leq \|c_i^\Theta\|_\infty + \|c_i^\Gamma\|_\infty = \|c_i^\Theta a_i^\Theta\|_\infty^{1/2} + \|c_i^\Gamma a_i^\Gamma\|_\infty^{1/2} \leq 2(2r)^{1/2} \tag{122}$$

and analogously $\|a_i^\Theta - a_i^\Gamma\|_\infty \leq 2(2r)^{1/2}$.
For $i \in I_2$ we need to slightly deviate from the proof of Theorem 3.1. We can skip the reparametrization step in (108)-(111) due to balancedness and need to distinguish three cases:
Case A.1: We have $\|c_i^\Gamma a_i^\Gamma\|_\infty \leq 2r$ which, together with (119), implies $\|c_i^\Theta a_i^\Theta\|_\infty \leq 4r$. Due to balancedness it follows that

$$\|c_i^\Theta - c_i^\Gamma\|_\infty, \|a_i^\Theta - a_i^\Gamma\|_\infty \leq 4r^{1/2}. \tag{123}$$

Case A.2: We have $\|c_i^\Theta a_i^\Theta\|_\infty \leq 2r$ which, together with (119), implies $\|c_i^\Gamma a_i^\Gamma\|_\infty \leq 4r$. Again it follows that

$$\|c_i^\Theta - c_i^\Gamma\|_\infty, \|a_i^\Theta - a_i^\Gamma\|_\infty \leq 4r^{1/2}. \tag{124}$$

Case B: We have $\|c_i^\Theta a_i^\Theta\|_\infty > 2r$ and $\|c_i^\Gamma a_i^\Gamma\|_\infty > 2r$. Due to the definition of I_2 there exists $e_i \in \mathbb{R}^d$, $\lambda_i^\Gamma, \lambda_i^\Theta \in (0,\infty)$ with $\|e_i\|_\infty = 1$, $a_i^\Theta = \lambda_i^\Theta e_i$, and $a_i^\Gamma = \lambda_i^\Gamma e_i$. As in (115) we obtain that

$$\begin{aligned}\|a_i^\Theta - a_i^\Gamma\|_\infty &= \|e_i\|_\infty|\lambda_i^\Theta - \lambda_i^\Gamma| \leq |(\lambda_i^\Theta)^2 - (\lambda_i^\Gamma)^2|^{1/2} \\ &= |\|c_i^\Theta\|_\infty\|a_i^\Theta\|_\infty - \|c_i^\Gamma\|_\infty\|a_i^\Gamma\|_\infty|^{1/2} \\ &\leq \|c_i^\Theta a_i^\Theta - c_i^\Gamma a_i^\Gamma\|_\infty^{1/2} \leq (2r)^{1/2}.\end{aligned} \tag{125}$$

Let now w.l.o.g. $\|a_i^\Gamma\|_\infty \geq \|a_i^\Theta\|_\infty$ (otherwise we switch their roles in the following) which implies that $\lambda_i^\Gamma = \Delta_i + \lambda_i^\Theta$ with $\Delta_i = \lambda_i^\Gamma - \lambda_i^\Theta \geq 0$. Then it holds that

$$\begin{aligned}\|c_i^\Theta - c_i^\Gamma\|_\infty &= \frac{\|c_i^\Theta a_i^\Gamma - c_i^\Gamma a_i^\Gamma\|_\infty}{\|a_i^\Gamma\|_\infty} \leq \frac{\|c_i^\Theta a_i^\Gamma - c_i^\Theta a_i^\Theta\|_\infty + \|c_i^\Theta a_i^\Theta - c_i^\Gamma a_i^\Gamma\|_\infty}{\|a_i^\Gamma\|_\infty} \\ &\leq \frac{\|c_i^\Theta\|_\infty|\lambda_i^\Gamma - \lambda_i^\Theta| + 2r}{\lambda_i^\Gamma} = \frac{\lambda_i^\Theta\Delta_i + 2r}{\Delta_i + \lambda_i^\Theta} \\ &= \frac{(2r)^{1/2}(\Delta_i + \lambda_i^\Theta) - (\lambda_i^\Theta - (2r)^{1/2})((2r)^{1/2} - \Delta_i)}{\Delta_i + \lambda_i^\Theta} \leq (2r)^{1/2}.\end{aligned} \tag{126}$$

The last step holds due to (125) and the balancedness of Θ which ensure that

$$\lambda_i^\Theta = \|c_i^\Theta a_i^\Theta\|_\infty^{1/2} > (2r)^{1/2} \geq |\lambda_i^\Theta - \lambda_i^\Gamma| = \Delta_i. \tag{127}$$

This completes the proof. □

A.4 Section 4

A.4.1 Additional Material

Lemma A.7 (Inverse stability for fixed weight vectors)**.** *Let $N = (d, m, D) \in \mathbb{N}^3$, let $A = [a_1|\dots|a_m]^T \in \mathbb{R}^{m\times d}$ with*

$$\frac{a_i}{\|a_i\|_\infty} \neq \frac{a_j}{\|a_j\|_\infty} \quad \textit{and} \quad (a_i)_{d-1}, (a_i)_d > 0 \tag{128}$$

for all $i \in [m], j \in [m] \setminus \{i\}$*, and define*

$$\mathcal{N}_N^A := \left\{\Gamma \in \mathcal{N}_N \colon a_i^\Gamma = \lambda_i a_i \text{ with } \lambda_i \in (0,\infty) \text{ and } \|c_i^\Gamma\|_\infty = \|a_i^\Gamma\|_\infty \text{ for all } i \in [m]\right\}. \quad (129)$$

Then for every $B \in (0,\infty)$ *there is* $C_B \in (0,\infty)$ *such that we have uniform* $(C_B, 1/2)$ *inverse stability w.r.t.* $\|\cdot\|_{L^\infty((-B,B)^d)}$*. That is, for all* $\Gamma \in \mathcal{N}_N^A$ *and* $g \in \mathcal{R}(\mathcal{N}_N^A)$ *there exists a parametrization* $\Phi \in \mathcal{N}_N^A$ *with*

$$\mathcal{R}(\Phi) = g \quad \text{and} \quad \|\Phi - \Gamma\|_\infty \leq C_B \|g - \mathcal{R}(\Gamma)\|_{L^\infty((-B,B)^d)}^{\frac{1}{2}}. \quad (130)$$

Proof. Note that the non-zero angle between the hyperplanes given by the weight vectors $(a_i)_{i=1}^m$ establishes that the minimal perimeter inside each linear region intersected with $(-B,B)^d$ is lower bounded. As the realization is linear on each region, this implies the existence of a constant $C_B' \in (0,\infty)$, such that for every $\Theta \in \mathcal{N}_N^A$ it holds that

$$|\mathcal{R}(\Theta)|_{W^{1,\infty}} \leq C_B' \|\mathcal{R}(\Theta)\|_{L^\infty((-B,B)^d)}. \quad (131)$$

Now note that for $\mathcal{N}_N^A$ we can get the same uniform $(4, 1/2)$ inverse stability result w.r.t. $|\cdot|_{W^{1,\infty}}$ as in Theorem 3.3 by choosing π to be the identity in (118). Together with (131) this implies the claim. □

VII Learning ReLU Networks to High Uniform Accuracy is Intractable

Comments

Conference: Poster presentation at ICLR 2023.
Code: github.com/juliusberner/theory2practice
E-Print: arXiv:2205.13531[cs.LG]

Contribution

The initial idea was drafted in equal parts by Philipp Grohs and Felix Voigtlaender. Julius Berner further developed the theory and contributed to the concept and writing of the paper. He also implemented and conducted the numerical experiments.

Bibliographic Information

Berner, Julius, Philipp Grohs, and Felix Voigtlaender (2023). "Learning ReLU networks to high uniform accuracy is intractable." In: *International Conference on Learning Representations*.

LEARNING RELU NETWORKS TO HIGH UNIFORM ACCURACY IS INTRACTABLE

Julius Berner[1,*], **Philipp Grohs**[1,2,3,*], and **Felix Voigtlaender**[4,*]

[1]Faculty of Mathematics, University of Vienna, Austria
[2]Research Network Data Science @ Uni Vienna, University of Vienna, Austria
[3]RICAM, Austrian Academy of Sciences, Austria
[4]Mathematical Institute for Machine Learning and Data Science (MIDS), Catholic University of Eichstätt-Ingolstadt, Germany
*All authors contributed equally

ABSTRACT

Statistical learning theory provides bounds on the necessary number of training samples needed to reach a prescribed accuracy in a learning problem formulated over a given target class. This accuracy is typically measured in terms of a generalization error, that is, an expected value of a given loss function. However, for several applications — for example in a security-critical context or for problems in the computational sciences — accuracy in this sense is not sufficient. In such cases, one would like to have guarantees for high accuracy on every input value, that is, with respect to the uniform norm. In this paper we precisely quantify the number of training samples needed for any conceivable training algorithm to guarantee a given uniform accuracy on any learning problem formulated over target classes containing (or consisting of) ReLU neural networks of a prescribed architecture. We prove that, under very general assumptions, the minimal number of training samples for this task scales exponentially both in the depth and the input dimension of the network architecture.

1 INTRODUCTION

The basic goal of supervised learning is to determine a function[1] $u : [0,1]^d \to \mathbb{R}$ from (possibly noisy) samples $(u(x_1), \dots, u(x_m))$. As the function u can take arbitrary values between these samples, this problem is, of course, not solvable without any further information on u. In practice, one typically leverages domain knowledge to estimate the structure and regularity of u a priori, for instance, in terms of symmetries, smoothness, or compositionality. Such additional information can be encoded via a suitable *target class* $U \subset C([0,1]^d)$ that u is known to be a member of. We are interested in the optimal accuracy for reconstructing u that can be achieved by any algorithm which utilizes m point samples. To make this mathematically precise, we assume that this accuracy is measured by a norm $\|\cdot\|_Y$ of a suitable Banach space $Y \supset U$. Formally, an algorithm can thus be described by a map $A : U \to Y$ that can query the function u at m points x_i and that outputs a function $A(u)$ with $A(u) \approx u$ (see Section 2.1 for a precise definition that incorporates adaptivity and stochasticity). We will be interested in *upper and lower bounds* on the accuracy that can be reached by any such algorithm — equivalently, we are interested in the minimal number m of point samples needed for any algorithm to achieve a given accuracy ε for every $u \in U$. This m would then establish a fundamental benchmark on the sample complexity (and the algorithmic complexity) of learning functions in U to a given accuracy.

The choice of the Banach space Y — in other words *how we measure accuracy* — is very crucial here. For example, statistical learning theory provides upper bounds on the optimal accuracy in terms of an expected loss, i.e., with respect to $Y = L^2([0,1]^d, \mathrm{d}\mathbb{P})$, where $\mathbb{P}$ is a (generally unknown)

[1]In what follows, the input domain $[0,1]^d$ could be replaced by more general domains (for example Lipschitz domains) without any change in the later results. The unit cube $[0,1]^d$ is merely chosen for concreteness.

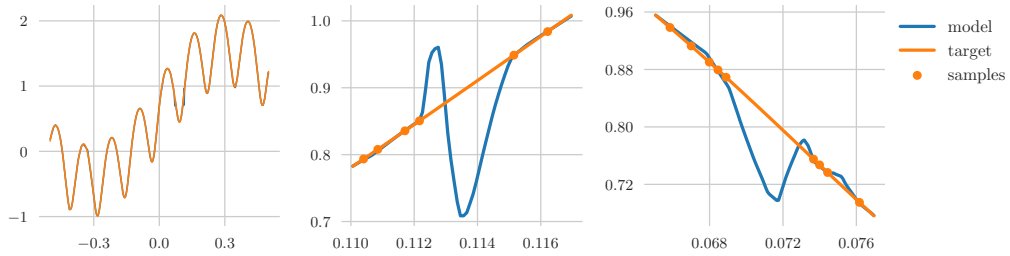


Figure 1: Even though the training of neural networks from data samples may achieve a small error *on average*, there are typically regions in the input space where the pointwise error is large. The target function in this plot is given by $x \mapsto \log(\sin(50x) + 2) + \sin(5x)$ (based on Adcock & Dexter, 2021) and the model is a feed-forward neural network. It is trained on $m = 1000$ uniformly distributed samples according to the hyperparameters in Tables 1 and 2 and achieves final L^1 and L^∞ errors of $2.8 \cdot 10^{-3}$ and 0.19, respectively. The middle and right plots are zoomed versions of the left plot.

data generating distribution (Devroye et al., 2013; Shalev-Shwartz & Ben-David, 2014; Mohri et al., 2018; Kim et al., 2021). This offers a powerful approach to ensure a small *average* reconstruction error. However, there are many important scenarios where such bounds on the accuracy are not sufficient and one would like to obtain an approximation of u that is close to u not only on average, but that can be guaranteed to be close *for every* $x \in [0,1]^d$. This includes several applications in the sciences, for example in the context of the numerical solution of partial differential equations (Raissi et al., 2019; Han et al., 2018; Richter & Berner, 2022), any security-critical application, for example, facial ID authentication schemes (Guo & Zhang, 2019), as well as any application with a *distribution-shift*, i.e., where the data generating distribution is different from the distribution in which the accuracy is measured (Quiñonero-Candela et al., 2008). Such applications can only be efficiently solved if there exists an efficient algorithm A that achieves *uniform accuracy*, i.e., a small error $\sup_{u \in U} \|u - A(u)\|_{L^\infty([0,1]^d)}$ with respect to the uniform norm given by $Y = L^\infty([0,1]^d)$, i.e., $\|f\|_{L^\infty([0,1]^d)} := \operatorname{esssup}_{x \in [0,1]^d} |f(x)|$.

Inspired by recent successes of *deep learning* across a plethora of tasks in machine learning (LeCun et al., 2015) and also increasingly the sciences (Jumper et al., 2021; Pfau et al., 2020), we will be particularly interested in the case where the target class U consists of — or contains — realizations of (feed-forward) neural networks of a specific architecture[2]. Neural networks have been proven and observed to be extremely powerful in terms of their expressivity, that is, their ability to accurately approximate large classes of complicated functions with only relatively few parameters (Elbrächter et al., 2021; Berner et al., 2022). However, it has also been repeatedly observed that the *training* of neural networks (e.g., fitting a neural network to data samples) to high *uniform* accuracy presents a big challenge: conventional training algorithms (such as SGD and its variants) often find neural networks that perform well on average (meaning that they achieve a small generalization error), but there are typically some regions in the input space where the error is large (Fiedler et al., 2023); see Figure 1 for an illustrative example. This phenomenon has been systematically studied on an empirical level by Adcock & Dexter (2021). It is also at the heart of several observed instabilities in the training of deep neural networks, including *adversarial examples* (Szegedy et al., 2013; Goodfellow et al., 2015) or so-called *hallucinations* emerging in generative modeling, e.g., tomographic reconstructions (Bhadra et al., 2021) or machine translation (Müller et al., 2020).

Note that additional knowledge on the target functions could potentially help circumvent these issues, see Remark 1.3. However, for many applications, it is not possible to precisely describe the regularity of the target functions. We thus analyze the case where no additional information is given besides the fact that one aims to recover a (unknown) neural network of a specified architecture and regularization from given samples – i.e., we assume that U contains a class of neural networks of a given architecture, subject to various regularization methods. This is satisfied in several applications of interest, e.g., *model extraction attacks* (Tramèr et al., 2016; He et al., 2022) and *teacher-student* settings (Mirzadeh et al., 2020; Xie et al., 2020). It is also in line with standard settings in the *statistical query literature*,

[2]By *architecture* we mean the number of layers L, as well as the number of neurons in each layer.

in *neural network identification*, and in statistical learning theory (Anthony & Bartlett, 1999; Mohri et al., 2018), see Section 1.1.

For such settings we can rigorously show that learning a class of neural networks is prone to instabilities. Specifically, any conceivable learning algorithm (in particular, any version of SGD), which recovers the neural network to high uniform accuracy, needs intractably many samples.

Theorem 1.1. *Suppose that U contains all neural networks with d-dimensional input, ReLU activation function, L layers of width up to $3d$, and coefficients bounded by c in the ℓ^q norm. Assume that there exists an algorithm that reconstructs all functions in U to uniform accuracy ε from m point samples. Then, we have*

$$m \geqslant \left(\frac{\Omega}{32d}\right)^d \cdot \varepsilon^{-d}, \quad \textit{where} \quad \Omega := \begin{cases} \frac{1}{8\cdot 3^{2/q}} \cdot c^L \cdot d^{1-\frac{2}{q}} & \textit{if } q \leqslant 2 \\ \frac{1}{48} \cdot c^L \cdot (3d)^{(L-1)(1-\frac{2}{q})} & \textit{if } q \geqslant 2. \end{cases}$$

Theorem 1.1 is a special case of Theorem 2.2 (covering $Y = L^p([0,1]^d)$ for all $p \in [1,\infty]$, as well as network architectures with arbitrary width) which will be stated and proven in Section 2.3.

To give a concrete example, we consider the problem of learning neural networks with ReLU activation function, L layers of width at most $3d$, and coefficients bounded by c to uniform accuracy $\varepsilon = 1/1024$. According to our results we would need at least

$$m \geqslant 2^d \cdot c^{dL} \cdot (3d)^{d(L-2)}$$

many samples — *the sample complexity thus depends exponentially on the input dimension d, the network width, and the network depth*, becoming intractable even for moderate values of d, c, L (for $d = 15$, $c = 2$, and $L = 7$, the sample size m would already have to exceed the estimated number of atoms in our universe). If, on the other hand, reconstruction only with respect to the L^2 norm were required, standard results in statistical learning theory (see, for example, Berner et al., 2020) show that m only needs to depend polynomially on d. We conclude that uniform reconstruction is vastly harder than reconstruction with respect to the L^2 norm and, in particular, intractable. Our results are further corroborated by numerical experiments presented in Section 3 below.

Remark 1.2. *For other target classes U, uniform reconstruction is tractable (i.e., the number of required samples for recovery does not massively exceed the number of parameters defining the class). A simple example are univariate polynomials of degree less than m which can be exactly determined from m samples. One can show similar results for sparse multivariate polynomials using techniques from the field of compressed sensing (Rauhut, 2007). Further, one can show that approximation rates in suitable reproducing kernel Hilbert spaces with bounded kernel can be realized using point samples with respect to the uniform norm (Pozharska & Ullrich, 2022). Our results uncover an opposing behavior of neural network classes: There exist functions that can be arbitrarily well approximated (in fact, exactly represented) by small neural networks, but these representations cannot be inferred from samples. Our results are thus highly specific to classes of neural networks.*

Remark 1.3. *Our results do not rule out the possibility that there exist training algorithms for neural networks that achieve high accuracy on some restricted class of target functions, if the knowledge about the target class can be incorporated into the algorithm design. For example, if it were known that the target function can be efficiently approximated by polynomials one could first compute an approximating polynomial (using polynomial regression which is tractable) and then represent the approximating polynomial by a neural network. The resulting numerical problem would however be very different from the way deep learning is used in practice, since most neural network coefficients (namely those corresponding to the approximating polynomial) would be fixed a priori. Our results apply to the situation where such additional information on the target class U is not available and no problem specific knowledge is incorporated into the algorithm design besides the network architecture and regularization procedure.*

We also complement the lower bounds of Theorem 1.1 with corresponding upper bounds.

Theorem 1.4. *Suppose that U consists of all neural networks with d-dimensional input, ReLU activation function, L layers of width at most B, and coefficients bounded by c in the ℓ^q norm. Then, there exists an algorithm that reconstructs all functions in U to uniform accuracy ε from m point samples with*

$$m \leqslant C^d \cdot \varepsilon^{-d}, \quad \textit{where} \quad C := \begin{cases} \sqrt{d} \cdot c^L & \textit{if } q \leqslant 2 \\ d^{1-\frac{1}{q}} \cdot c^L \cdot B^{(L-1)(1-\frac{2}{q})} & \textit{if } q \geqslant 2. \end{cases}$$

Theorem 1.4 follows from Theorem 2.4 that will be stated in Section 2.4. We refer to Remark B.4 for a discussion of the gap between the upper and lower bounds.

Remark 1.5. *Our setting allows for an algorithm to choose the sample points $(x_1, \dots, x_m)$ in an adaptive way for each $u \in U$; see Section 2.1 for a precise definition of the class of adaptive (possibly randomized) algorithms. This implies that even a very clever sampling strategy (as would be employed in active learning) cannot break the bounds established in this paper.*

Remark 1.6. *Our results also shed light on the impact of different regularization methods. While picking a stronger regularizer (e.g., a small value of q) yields quantitative improvements (in the sense of a smaller Ω), the sample size m required for approximation in L^∞ can still increase exponentially with the input dimension d. However, this scaling is only visible for very small ε.*

1.1 Related Work

Several other works have established "hardness" results for neural network training. For example, the seminal works by Blum & Rivest (1992); Vu (1998) show that for certain architectures the training process can be NP-complete. By contrast, our results do not directly consider algorithm runtime at all; our results are stronger in the sense of showing that even if it were possible to efficiently learn a neural network from samples, the necessary number of data points would be too large to be tractable.

We also want to mention a series of hardness results in the setting of *statistical query* (SQ) algorithms, see, e.g., Chen et al. (2022); Diakonikolas et al. (2020); Goel et al. (2020b); Reyzin (2020); Song et al. (2017). For instance, Chen et al. (2022) shows that any SQ algorithm capable of learning ReLU networks with two hidden layers and width $\mathrm{poly}(d)$ up to L^2 error $1/\mathrm{poly}(d)$ must use a number of samples that scales *superpolynomially* in d, or must use SQ queries with tolerance smaller than the reciprocal of any polynomial in d. In such SQ algorithms, the learner has access to an oracle that produces approximations (potentially corrupted by *adversarial noise*) of certain expectations $\mathbb{E}[h(X, u(X))]$, where u is the unknown function to be learned, X is a random variable representing the data, and h is a function chosen by the learner (potentially subject to some restrictions, e.g. Lipschitz continuity). The possibility of the oracle to inject adversarial (instead of just stochastic) noise into the learning procedure — which does not entirely reflect the typical mathematical formulation of learning problems — is crucial for several of these results. We also mention that due to this possibility of adversarial noise, not every gradient-based optimization method (for instance, SGD) is strictly speaking an SQ algorithm; see also the works by Goel et al. (2020a, Page 3) and Abbe et al. (2021) for a more detailed discussion.

There also exist hardness results for learning algorithms based on *label queries* (i.e., noise-free point samples), which constitutes a setting similar to ours. More precisely, Chen et al. (2022) show that ReLU neural networks with constant depth and polynomial size constraints are not efficiently learnable up to a small squared loss with respect to a Gaussian distribution. However, the existing hardness results are in terms of *runtime* of the algorithm and are contingent on several (difficult and unproven) conjectures from the area of cryptography (the decisional Diffie-Hellmann assumption or the "Learning with Errors" assumption); the correctness of these conjectures in particular would imply that $\mathrm{P} \neq \mathrm{NP}$. By contrast, our results are completely free of such assumptions and show that the considered problem is *information-theoretically* hard, not just computationally.

As already hinted at in the introduction, our results further extend the broad literature on statistical learning theory (Anthony & Bartlett, 1999; Vapnik, 1999; Cucker & Smale, 2002b; Bousquet et al., 2003; Vapnik, 2013; Mohri et al., 2018). Specifically, we provide fully explicit upper and lower bounds on the *sample complexity* of (regularized) neural network *hypothesis classes*. In the context of PAC learning, we analyze the realizable case, where the target function is contained in the hypothesis class (Mohri et al., 2018, Theorem 3.20). Contrary to standard results, we do not pose any assumptions, such as IID, on the data distribution, and even allow for adaptive sampling. Moreover, we analyze the complexity for all L^p norms with $p \in [1, \infty]$, whereas classical results mostly deal with the squared loss. As an example of such classical results, we mention that (bounded) hypothesis classes with finite pseudodimension D can be learned to squared L^2 loss ε with $\mathcal{O}(D\varepsilon^{-2})$ point samples; see e.g., Mohri et al. (2018, Theorem 11.8). Bounds for the pseudodimension of neural networks are readily available in the literature; see e.g., Bartlett et al. (2019). These bounds imply that learning ReLU networks in L^2 is tractable, in contrast to the L^∞ setting.

Another related area is the identification of (equivalence classes of) neural network parameters from their input-output maps. While most works focus on scenarios where one has access to an infinite number of queries (Fefferman & Markel, 1993; Vlačić & Bölcskei, 2022), there are recent results employing only finitely many samples (Rolnick & Kording, 2020; Fiedler et al., 2023). Robust identification of the neural network parameters is sufficient to guarantee uniform accuracy, but it is not a necessary condition. Specifically, proximity of input-output maps does not necessarily imply proximity of corresponding neural network parameters (Berner et al., 2019). More generally, our results show that efficient identification from samples cannot be possible unless (as done in the previously mentioned works) further prior information is incorporated. In the same spirit, this restricts the applicability of model extraction attacks, such as *model inversion* or *evasion attacks* (Tramèr et al., 2016; He et al., 2022).

Our results are most closely related to recent results by Grohs & Voigtlaender (2021) where target classes consisting of neural network approximation spaces are considered. The results of Grohs & Voigtlaender (2021), however, are purely asymptotic. Since the asymptotic behavior incurred by the rate is often only visible for very fine accuracies, the results of Grohs & Voigtlaender (2021) cannot be applied to obtain concrete lower bounds on the required sample size. Our results are completely explicit in all parameters and readily yield practically relevant bounds. They also elucidate the role of adaptive sampling and different regularization methods.

1.2 Notation

For $d \in \mathbb{N}$, we denote by $C([0,1]^d)$ the space of continuous functions $f\colon [0,1]^d \to \mathbb{R}$. For a finite set I and $(a_i)_{i\in I} \in \mathbb{R}^I$, we write $\sum_{i\in I} a_i \coloneqq \frac{1}{|I|}\sum_{i\in I} a_i$. For $m \in \mathbb{N}$, we write $[m] \coloneqq \{1,\dots,m\}$. For $A \subset \mathbb{R}^d$, we denote by A° the set of interior points of A. For any subset A of a vector space V, any $c \in \mathbb{R}$, and any $y \in V$, we further define $y + c\cdot A \coloneqq \{y + ca\colon a \in A\}$. For a matrix $W \in \mathbb{R}^{n\times k}$ and $q \in [1,\infty)$, we write $\|W\|_{\ell^q} \coloneqq \big(\sum_{i,j} |W_{i,j}|^q\big)^{1/q}$, and for $q = \infty$ we write $\|W\|_{\ell^\infty} \coloneqq \max_{i,j} |W_{i,j}|$. For vectors $b \in \mathbb{R}^n$, we use the analogously defined notation $\|b\|_{\ell^q}$.

2 Main Results

This section contains our main theoretical results. We introduce the considered classes of algorithms in Section 2.1 and target classes in Section 2.2. Our main lower and upper bounds are formulated and proven in Section 2.3 and Section 2.4, respectively.

2.1 Adaptive (randomized) algorithms based on point samples

As described in the introduction, our goal is to analyze how well one can recover an unknown function u from a target class U in a Banach space Y based on point samples. This is one of the main problems in *information-based complexity* (Traub, 2003), and in this section we briefly recall the most important related notions.

Given $U \subset C([0,1]^d) \cap Y$ for a Banach space Y, we say that a map $A : U \to Y$ is an *adaptive deterministic method using* $m \in \mathbb{N}$ *point samples* if there are $f_1 \in [0,1]^d$ and mappings

$$f_i : \big([0,1]^d\big)^{i-1} \times \mathbb{R}^{i-1} \to [0,1]^d, \quad i = 2,\dots,m, \quad \text{and} \quad Q : \big([0,1]^d\big)^m \times \mathbb{R}^m \to Y$$

such that for every $u \in U$, using the point sequence $\mathbf{x}(u) = (x_1,\dots,x_m) \subset [0,1]^d$ defined as

$$x_1 = f_1, \quad x_i = f_i(x_1,\dots,x_{i-1},u(x_1),\dots,u(x_{i-1})), \quad i = 2,\dots,m, \tag{1}$$

the map A is of the form $A(u) = Q(x_1,\dots,x_m,u(x_1),\dots,u(x_m)) \in Y$.

The set of all deterministic methods using m point samples is denoted by $\mathrm{Alg}_m(U,Y)$. In addition to such deterministic methods, we also study randomized methods defined as follows: A tuple $(\mathbf{A},\mathbf{m})$ is called an *adaptive random method using* $m \in \mathbb{N}$ *point samples on average* if $\mathbf{A} = (A_\omega)_{\omega\in\Omega}$ where $(\Omega,\mathcal{F},\mathbb{P})$ is a probability space, and where $\mathbf{m} : \Omega \to \mathbb{N}$ is such that the following conditions hold:

1. $\mathbf{m}$ is measurable, and $\mathbb{E}[\mathbf{m}] \leqslant m$;
2. $\forall\, u \in U :\ \omega \mapsto A_\omega(u)$ is measurable with respect to the Borel σ-algebra on Y;

3. $\forall\, \omega \in \Omega:\ A_\omega \in \mathrm{Alg}_{\mathbf{m}(\omega)}(U, Y)$.

The set of all random methods using m point samples on average will be denoted by $\mathrm{Alg}_m^{MC}(U, Y)$, since such methods are sometimes called *Monte-Carlo* (MC) algorithms.

For a target class U, we define the *optimal (randomized) error* as

$$\mathrm{err}_m^{MC}(U, Y) := \inf_{(\mathbf{A},\mathbf{m})\in \mathrm{Alg}_m^{MC}(U,Y)} \sup_{u\in U} \mathbb{E}\left[\|u - A_\omega(u)\|_Y\right]. \tag{2}$$

We note that $\mathrm{Alg}_m(U, Y) \subset \mathrm{Alg}_m^{MC}(U, Y)$, since each deterministic method can be interpreted as a randomized method over a trivial probability space.

2.2 Neural network classes

We will be concerned with target classes related to ReLU neural networks. These will be defined in the present subsection. Let $\varrho : \mathbb{R} \to \mathbb{R}$, $\varrho(x) = \max\{0, x\}$, be the *ReLU activation function*. Given a *depth* $L \in \mathbb{N}$, an *architecture* $(N_0, N_1, \ldots, N_L) \in \mathbb{N}^{L+1}$, and *neural network coefficients*

$$\Phi = \left((W^i, b^i)\right)_{i=1}^L \in \bigtimes_{i=1}^L \left(\mathbb{R}^{N_i\times N_{i-1}} \times \mathbb{R}^{N_i}\right),$$

we define their *realization* $R(\Phi) \in C(\mathbb{R}^{N_0}, \mathbb{R}^{N_L})$ as

$$R(\Phi) := \phi^L \circ \varrho \circ \phi^{L-1} \circ \cdots \circ \varrho \circ \phi^1$$

where ϱ is applied componentwise and $\phi^i \colon \mathbb{R}^{N_{i-1}} \to \mathbb{R}^{N_i}$, $x \mapsto W^i x + b^i$, for $i \in [L]$. Given $c > 0$ and $q \in [1, \infty]$, define the class

$$\mathcal{H}^q_{(N_0,\ldots,N_L),c} := \left\{R(\Phi) :\ \Phi \in \bigtimes_{i=1}^L \left(\mathbb{R}^{N_i\times N_{i-1}} \times \mathbb{R}^{N_i}\right) \text{ and } \|\Phi\|_{\ell^q} \leqslant c\right\},$$

where $\|\Phi\|_{\ell^q} := \max_{1\leqslant i\leqslant L} \max\{\|W^i\|_{\ell^q}, \|b^i\|_{\ell^q}\}$.

To study target classes related to neural networks, the following definition will be useful.

Definition 2.1. *Let $U, \mathcal{H} \subset C([0, 1]^d)$. We say that U contains a copy of $\mathcal{H}$, attached to $u_0 \in U$ with constant $c_0 \in (0, \infty)$, if $u_0 + c_0 \cdot \mathcal{H} \subset U$.*

2.3 Lower bound

The following result constitutes the main result of the present paper. Theorem 1.1 readily follows from it as a special case.

Theorem 2.2. *Let $L \in \mathbb{N}_{\geqslant 3}$, $d, B \in \mathbb{N}$, $p, q \in [1, \infty]$, and $c \in (0, \infty)$. Suppose that the target class $U \subset C([0, 1]^d)$ contains a copy of $\mathcal{H}^q_{(d,B\ldots,B,1),c}$ with constant $c_0 \in (0, \infty)$, where the B in $(d, B, \ldots, B, 1)$ appears $L - 1$ times. Then, for any $s \in \mathbb{N}$ with $s \leqslant \min\left\{\frac{B}{3}, d\right\}$ we have*

$$\mathrm{err}_m^{MC}(U, L^p([0, 1]^d)) \geqslant c_0 \cdot \frac{\Omega}{(32s)^{1+\frac{s}{p}}} \cdot m^{-\frac{1}{p}-\frac{1}{s}},$$

where

$$\Omega := \begin{cases} \frac{1}{8\cdot 3^{2/q}} \cdot c^L \cdot s^{1-\frac{2}{q}} & \text{if } q \leqslant 2 \\ \frac{1}{48} \cdot c^L \cdot B^{(L-1)(1-\frac{2}{q})} & \text{if } q \geqslant 2. \end{cases}$$

Proof. This follows by combining Theorem A.5 with Lemmas A.2 and A.3 in the appendix. □

Remark 2.3. *For $p \ll \infty$, the bound from above does not necessarily imply that an intractable number of training samples is needed. This is a reflection of the fact that efficient learning is possible (at least if one only considers the number of training samples and not the runtime of the algorithm) in this regime. Indeed, it is well-known in statistical learning theory that one obtains learning bounds based on the entropy numbers (w.r.t. the L^∞ norm) of the class of target functions, when the error is measured in L^2, see, for instance, Cucker & Smale (2002a, Proposition 7). The ε-entropy numbers of a class of neural networks with L layers and w (bounded) weights scale linearly in w, L and*

logarithmically in $1/\varepsilon$, so that one gets tractable L^2 learning bounds. By interpolation for L^p norms (noting that in our case the target functions are bounded, so that the L^∞ reconstruction error is bounded, even though the decay with m is very bad), this also implies L^p learning bounds, but these get worse and worse as $p \to \infty$. We remark that these learning bounds are based on empirical risk minimization, which might be computationally infeasible (Vu, 1998); since our lower bounds should hold for any feasible algorithm (irrespective of its computational complexity), this means that one cannot expect to get an intractable lower bound for $p \ll \infty$ in our setting.

The idea of the proof of Theorem 2.2 (here only presented for $u_0 = 0$ and $s = d$, which implies that $B \geqslant 3d$) is as follows:

1. We first show (see Lemmas A.2 and A.3) that the neural network set $\mathcal{H}^q_{(d,B,\dots,B,1),c}$ contains a large class of "bump functions" of the form $\lambda \cdot \vartheta_{M,y}$. Here, $\vartheta_{M,y}$ is supported on the set $y + [-\frac{1}{M}, \frac{1}{M}]^d$ and satisfies $\|\vartheta_{M,y}\|_{L^p([0,1]^d)} \asymp M^{-d/p}$, where $M \in \mathbb{N}$ and $y \in [0,1]^d$ can be chosen arbitrarily; see Lemma A.1. The size of the scaling factor $\lambda = \lambda(M, c, q, d, L)$ depends crucially on the regularization parameters c and q. This is the main technical part of the proof, requiring to construct suitable neural networks adhering to the imposed ℓ^q restrictions on the weights for which λ is as big as possible.
2. If one learns using m points samples $x_1, \dots, x_m$ and if $M = \mathcal{O}(m^{1/d})$, then a volume packing argument shows that there exists $y \in [0,1]^d$ such that $\vartheta_{M,y}(x_i) = 0$ for all $i \in [m]$. This means that the learner cannot distinguish the function $\lambda \cdot \vartheta_{M,y} \in \mathcal{H}^q_{(d,B,\dots,B,1),c}$ from the zero function and will thus make an error of roughly $\|\lambda \cdot \vartheta_{M,y}\|_{L^p} \asymp \lambda \cdot M^{-d/p}$. This already implies the lower bound in Theorem 2.2 for the case of *deterministic* algorithms.
3. To get the lower bound for *randomized* algorithms using m point samples on average, we employ a technique from information-based complexity (see, e.g., Heinrich, 1994): We again set $M = \mathcal{O}(m^{1/d})$ and define $(y_\ell)_{\ell \in [M/2]^d}$ as the nodes of a uniform grid on $[0,1]^d$ with width $2/M$. Using a volume packing argument, we then show that for any choice of m sampling points $x_1, \dots, x_m$, "at least half of the functions ϑ_{M,y_ℓ} avoid all the sampling points", i.e., for at least half of the indices ℓ, it holds that $\vartheta_{M,y_\ell}(x_i) = 0$ for all $i \in [m]$. A learner using the samples $x_1, \dots, x_m$ can thus not distinguish between the zero function and $\lambda \cdot \vartheta_{M,y_\ell} \in \mathcal{H}^q_{(d,B,\dots,B,1),c}$ for at least half of the indices ℓ. Therefore, any *deterministic* algorithm will make an error of $\Omega(\lambda \cdot M^{-d/p})$ *on average with respect to* ℓ.
4. Since each randomized algorithm $\boldsymbol{A} = (A_\omega)_{\omega \in \Omega}$ is a collection of deterministic algorithms and since taking an average commutes with taking the expectation, this implies that any randomized algorithm will have an expected error of $\Omega(\lambda \cdot M^{-d/p})$ *on average with respect to* ℓ. This easily implies the stated bound.

As mentioned in the introduction, we want to emphasize that well-trained neural networks can indeed exhibit such bump functions, see Figure 1 and Adcock & Dexter (2021); Fiedler et al. (2023).

2.4 Upper bound

In this section we present our main upper bound, which directly implies the statement of Theorem 1.4.

Theorem 2.4. *Let $L, d \in \mathbb{N}$, $q \in [1, \infty]$, $c \in (0, \infty)$, and $N_1, \dots, N_{L-1} \in \mathbb{N}$. Then, we have*

$$\operatorname{err}_m^{MC}\big(\mathcal{H}^q_{(d,N_1,\dots,N_{L-1},1),c}, L^\infty([0,1]^d)\big) \leqslant \begin{cases} \sqrt{d} \cdot c^L \cdot m^{-\frac{1}{d}} & \text{if } q \leqslant 2 \\ \sqrt{d} \cdot c^L \cdot (\sqrt{d} \cdot N_1 \cdots N_{L-1})^{1-\frac{2}{q}} \cdot m^{-\frac{1}{d}} & \text{if } q \geqslant 2. \end{cases}$$

Proof. This follows by combining Lemmas B.2 and B.3 in the appendix. □

Let us outline the main idea of the proof. We first show that each neural network $R(\Phi) \in \mathcal{H}^q_{(N_0,\dots,N_L),c}$ is Lipschitz-continuous, where the Lipschitz constant can be conveniently bounded in terms of the parameters $N_0, \dots, N_L, c, q$, see Lemma B.2 in the appendix. In Lemma B.3, we then show that any function with moderate Lipschitz constant can be reconstructed from samples by piecewise constant interpolation.

3 NUMERICAL EXPERIMENTS

Having established fundamental bounds on the performance of any learning algorithm, we want to numerically evaluate the performance of commonly used deep learning methods. To illustrate our main result in Theorem 2.2, we estimate the error in (2) by a tractable approximation in a student-teacher setting. Specifically, we estimate the minimal error over neural network target functions ("teachers") $\widehat{U} \subset \mathcal{H}^q_{(d,N_1,\dots,N_{L-1},1),c}$ for deep learning algorithms $\widehat{A} \subset \mathrm{Alg}^{MC}_m(U, L^p)$ via Monte-Carlo sampling, i.e.,

$$\widehat{\mathrm{err}}_m\left(\widehat{U}, L^p; \widehat{A}\right) := \inf_{(\mathbf{A},\mathbf{m})\in\widehat{A}} \sup_{u\in\widehat{U}} \sum_{\omega\in\widehat{\Omega}} \left(\sum_{j\in[J]} \left(u(X_j) - A_\omega(u)(X_j)\right)^p\right)^{1/p}, \qquad (3)$$

where $(X_j)_{j=1}^J$ are independent evaluation samples uniformly distributed on[3] $[-0.5, 0.5]^d$ and $\widehat{\Omega}$ represents the seeds for the algorithms.

We obtain teacher networks $u \in \mathcal{H}^\infty_{(d,N_1,\dots,N_{L-1},1),c}$ by sampling their coefficients Φ componentwise according to a uniform distribution on $[-c, c]$. For every algorithm $(\mathbf{A}, \mathbf{m}) \in \widehat{A}$ and seed $\omega \in \widehat{\Omega}$ we consider point sequences $\mathbf{x}(u)$ uniformly distributed in $[-0.5, 0.5]^d$ with $\mathbf{m}(\omega) = m$. The corresponding point samples are used to train the coefficients of a neural network ("student") using the Adam optimizer (Kingma & Ba, 2015) with exponentially decaying learning rate. We consider input dimensions $d = 1$ and $d = 3$, for each of which we compute the error in (3) for 4 different sample sizes m over 40 teacher networks u. For each combination, we train student networks with 3 different seeds, 3 different widths, and 3 different batch-sizes. In summary, this yields $2 \cdot 4 \cdot 40 \cdot 3 \cdot 3 \cdot 3 = 8640$ experiments each executed on a single GPU. The precise hyperparameters can be found in Tables 1 and 3 in Appendix C.

Figure 2 shows that there is a clear gap between the errors $\widehat{\mathrm{err}}_m(\widehat{U}, L^p; \widehat{A})$ for $p \in \{1, 2\}$ and $p = \infty$. Especially in the one-dimensional case, the rate $\widehat{\mathrm{err}}_m(\widehat{U}, L^\infty; \widehat{A})$ w.r.t. the number of samples m also seems to stagnate at a precision that might be insufficient for certain applications. Figure 3 illustrates that the errors are caused by spikes of the teacher network which are not covered by any sample. Note that this is very similar to the construction in the proof of our main result, see Section 2.3.

In general, the rates worsen when considering more teacher networks $\widehat{U}$ and improve when considering further deep learning algorithms $\widehat{A}$, including other architectures or more elaborate training and sampling schemes. Note, however, that each setting needs to be evaluated for a number of teacher networks, sample sizes, and seeds. We provide an extensible implementation[4] in PyTorch (Paszke et al., 2019) featuring multi-node experiment execution and hyperparameter tuning using Ray Tune (Liaw et al., 2018), experiment tracking using Weights & Biases and TensorBoard, and flexible experiment configuration. Building upon our work, research teams with sufficient computational resources can provide further numerical evidence on an even larger scale.

4 DISCUSSION AND LIMITATIONS

Discussion. We derived fundamental upper and lower bounds for the number of samples needed for any algorithm to reconstruct an arbitrary function from a target class containing realizations of neural networks with ReLU activation function of a given architecture and subject to ℓ^q regularization constraints on the network coefficients, see Theorems 2.2 and 2.4. These bounds are completely explicit in the network architecture, the type of regularization, and the norm in which the reconstruction error is measured. We observe that our lower bounds are severely more restrictive if the error is measured in the uniform L^∞ norm rather than the (more commonly studied) L^2 norm. Particularly, learning a class of neural networks with ReLU activation function with moderately high accuracy in the L^∞ norm is intractable for moderate input dimensions, as well as network widths and depths. We anticipate that further investigations into the sample complexity of neural network classes can eventually contribute to a better understanding of possible circumstances under which it is possible to design reliable deep learning algorithms and help explain well-known instability phenomena such

[3] To have centered input data, we consider the hypercube $[-0.5, 0.5]^d$ in our experiments. Note that this does not change any of the theoretical results.

[4] The code can be found at `https://github.com/juliusberner/theory2practice`.

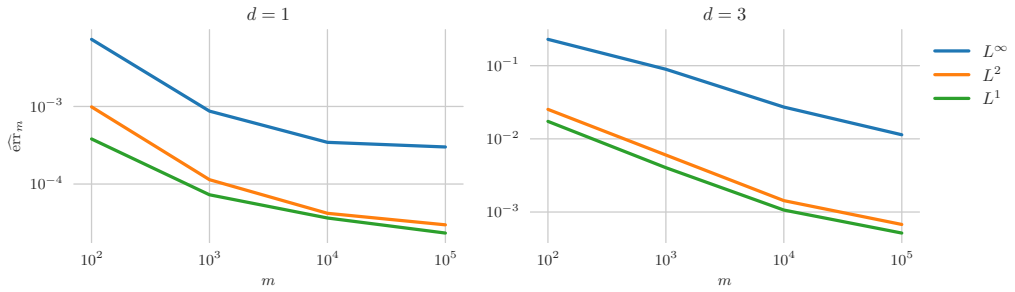


Figure 2: Evaluation of the error in (3) for $p \in \{1, 2, \infty\}$, input dimensions $d \in \{1, 3\}$, sample sizes $m \in \{10^2, 10^3, 10^4, 10^5\}$, and hyperparameters given in Tables 1 and 3.

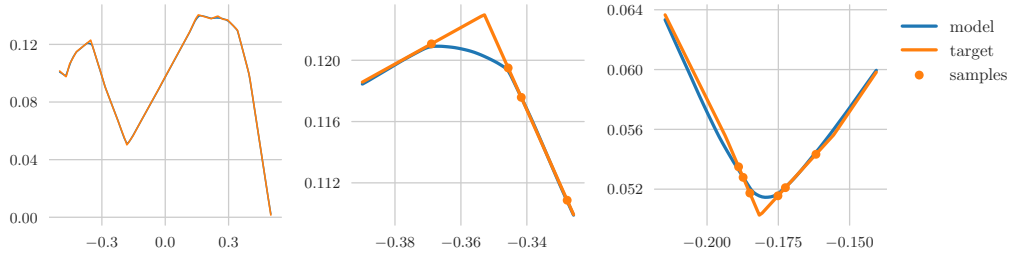


Figure 3: Target function ("teacher"), samples, and model of the deep learning algorithm ("student") attaining the min-max value in (3) for $m = 100$ and $p = \infty$ in the experiment depicted in Figure 2. The middle and right plots are zoomed versions of the left plot. The L^∞ error $(2.7 \cdot 10^{-3})$ is about one magnitude larger than the L^2 and L^1 errors $(3.9 \cdot 10^{-4}$ and $2.4 \cdot 10^{-4})$, which is caused by spikes of the teacher network between samples.

as adversarial examples. Such an understanding can be beneficial in assessing the potential and limitations of machine learning methods applied to security- and safety-critical scenarios.

Limitations and Outlook. We finally discuss some possible implications and also limitations of our work. First of all, our results are highly specific to neural networks with the ReLU activation function. We expect that obtaining similar results for other activation functions will require substantially new methods. We plan to investigate this in future work.

The explicit nature of our results reveal a discrepancy between the lower and upper bound, especially for high dimensions. We conjecture that both the current upper and lower bounds are not quite optimal. Determining to which extent one can tighten the bounds is an interesting open problem.

Our analysis is a worst-case analysis in the sense that we show that for any given algorithm A, there exists at least one u in our target class U on which A performs poorly. The question of whether this poor behavior is actually generic will be studied in future work. One way to establish such generic results could be to prove that our considered target classes contain copies of neural network realizations attached to many different u's.

Finally, we consider target classes U that contain all realizations of neural networks with a given architecture subject to different regularizations. This can be justified as follows: Whenever a deep learning method is employed to reconstruct a function u by representing it approximately by a neural network (without further knowledge about u), a natural minimal requirement is that the method should perform well if the sought function is in fact equal to a neural network. However, if additional problem information about u can be incorporated into the learning problem it may be possible to overcome the barriers shown in this work. The degree to which this is possible, as well as the extension of our results to other architectures, such as convolutional neural networks, transformers, and graph neural networks will be the subject of future work.

ACKNOWLEDGMENTS

The research of Julius Berner was supported by the Austrian Science Fund (FWF) under grant I3403-N32 and by the Vienna Science and Technology Fund (WWTF) under grant ICT19-041. The computational results presented have been achieved in part using the Vienna Scientific Cluster (VSC). Felix Voigtlaender acknowledges support by the DFG in the context of the Emmy Noether junior research group VO 2594/1-1.

REFERENCES

E. Abbe, P. Kamath, E. Malach, C. Sandon, and N. Srebro. On the power of differentiable learning versus PAC and SQ learning. *Advances in Neural Information Processing Systems*, 34:24340–24351, 2021.

Ben Adcock and Nick Dexter. The gap between theory and practice in function approximation with deep neural networks. *SIAM Journal on Mathematics of Data Science*, 3(2):624–655, 2021.

Martin Anthony and Peter L Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *The Journal of Machine Learning Research*, 20(1): 2285–2301, 2019.

Julius Berner, Dennis Maximilian Elbrächter, and Philipp Grohs. How degenerate is the parametrization of neural networks with the relu activation function? *Advances in Neural Information Processing Systems*, 32, 2019.

Julius Berner, Philipp Grohs, and Arnulf Jentzen. Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black–Scholes partial differential equations. *SIAM Journal on Mathematics of Data Science*, 2(3):631–657, 2020.

Julius Berner, Philipp Grohs, Gitta Kutyniok, and Philipp Petersen. *The Modern Mathematics of Deep Learning*, pp. 1–111. Cambridge University Press, 2022.

Sayantan Bhadra, Varun A Kelkar, Frank J Brooks, and Mark A Anastasio. On hallucinations in tomographic image reconstruction. *IEEE transactions on medical imaging*, 40(11):3249–3260, 2021.

Avrim L Blum and Ronald L Rivest. Training a 3-node neural network is NP-complete. *Neural Networks*, 5(1):117–127, 1992.

Edward K Blum and Leong Kwan Li. Approximation theory and feedforward networks. *Neural networks*, 4(4):511–515, 1991.

Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In *Summer School on Machine Learning*, pp. 169–207, 2003.

S. Chen, A. Gollakota, A. R. Klivans, and R. Meka. Hardness of noise-free learning for two-hidden-layer neural networks. *arXiv preprint arXiv:2202.05258*, 2022.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bull. Amer. Math. Soc. (N.S.)*, 39(1):1–49, 2002a. ISSN 0273-0979.

Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39(1):1–49, 2002b.

Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

I. Diakonikolas, D. Kane, and N. Zarifis. Near-optimal SQ lower bounds for agnostically learning halfspaces and ReLUs under Gaussian marginals. *Advances in Neural Information Processing Systems*, 33:13586–13596, 2020.

Dennis Elbrächter, Dmytro Perekrestenko, Philipp Grohs, and Helmut Bölcskei. Deep neural network approximation theory. *IEEE Transactions on Information Theory*, 67(5):2581–2623, 2021.

Charles Fefferman and Scott Markel. Recovering a feed-forward net from its output. *Advances in neural information processing systems*, 6, 1993.

Christian Fiedler, Massimo Fornasier, Timo Klock, and Michael Rauchensteiner. Stable recovery of entangled weights: Towards robust identification of deep neural networks from minimal samples. *Applied and Computational Harmonic Analysis*, 62:123–172, 2023.

G. B. Folland. *Real Analysis: Modern Techniques and Their Applications*. Pure and Applied Mathematics. John Wiley & Sons, second edition, 1999.

S. Goel, A. Gollakota, Z. Jin, S. Karmalkar, and A. Klivans. Superpolynomial lower bounds for learning one-layer neural networks using gradient descent. In *International Conference on Machine Learning*, pp. 3587–3596, 2020a.

S. Goel, A. Gollakota, and A. Klivans. Statistical-query lower bounds via functional gradients. *Advances in Neural Information Processing Systems*, 33:2147–2158, 2020b.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations*, 2015.

Philipp Grohs and Felix Voigtlaender. Proof of the theory-to-practice gap in deep learning via sampling complexity bounds for neural network approximation spaces. *arXiv preprint arXiv:2104.02746*, 2021.

Guodong Guo and Na Zhang. A survey on deep learning based face recognition. *Computer vision and image understanding*, 189:102805, 2019.

Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

Yingzhe He, Guozhu Meng, Kai Chen, Xingbo Hu, and Jinwen He. Towards security threats of deep learning systems: A survey. *IEEE Transactions on Software Engineering*, 48(5):1743–1770, 2022.

S. Heinrich. Random approximation in numerical analysis. In *Functional analysis*, volume 150 of *Lecture Notes in Pure and Appl. Math.*, pp. 123–171. Dekker, New York, 1994.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

Yongdai Kim, Ilsang Ohn, and Dongha Kim. Fast convergence rates of deep neural networks for classification. *Neural Networks*, 138:179–197, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*, 2015.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 5191–5198, 2020.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

Mathias Müller, Annette Rios Gonzales, and Rico Sennrich. Domain robustness in neural machine translation. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas*, pp. 151–164, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pp. 8024–8035, 2019.

D. Pfau, J. S. Spencer, A. G. D. G. Matthews, and W. M. C. Foulkes. Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Phys. Rev. Research*, 2:033429, Sep 2020.

Kateryna Pozharska and Tino Ullrich. A note on sampling recovery of multivariate functions in the uniform norm. *SIAM Journal on Numerical Analysis*, 60(3):1363–1384, 2022.

Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. MIT Press, 2008.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Holger Rauhut. Random sampling of sparse trigonometric polynomials. *Applied and Computational Harmonic Analysis*, 22(1):16–42, 2007.

L. Reyzin. Statistical queries and statistical algorithms: Foundations and applications. *arXiv preprint arXiv:2004.00557*, 2020.

Lorenz Richter and Julius Berner. Robust sde-based variational formulations for solving linear pdes via deep learning. In *International Conference on Machine Learning*, pp. 18649–18666, 2022.

David Rolnick and Konrad Kording. Reverse-engineering deep relu networks. In *International Conference on Machine Learning*, pp. 8178–8187, 2020.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

L. Song, S. Vempala, J. Wilmes, and B. Xie. On the complexity of learning neural networks. *Advances in neural information processing systems*, 30, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *25th USENIX security symposium (USENIX Security 16)*, pp. 601–618, 2016.

Joseph F Traub. Information-based complexity. In *Encyclopedia of Computer Science*, pp. 850–854. John Wiley & Sons, 2003.

Vladimir Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

Verner Vlačić and Helmut Bölcskei. Neural network identifiability for a family of sigmoidal nonlinearities. *Constructive Approximation*, 55(1):173–224, 2022.

V.H. Vu. On the infeasibility of training neural networks with small mean-squared error. *IEEE Transactions on Information Theory*, 44(7):2892–2900, 1998.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10687–10698, 2020.

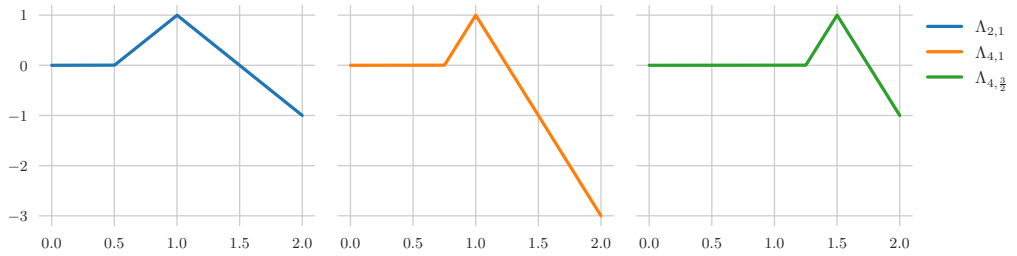


Figure 4: Plots of the function $\Lambda_{M,\sigma}$ in Equation (4) for $(M,\sigma) \in \{(2,1),(4,1),(4,\frac{3}{2})\}$.

A Proof of the lower bound in Section 2.3

A.1 Construction of hat functions implemented by ReLU networks

For $d \in \mathbb{N}$, $M > 0$, $\sigma \in \mathbb{R}$, $s \in [d]$, and $y \in \mathbb{R}^d$, define

$$\Lambda_{M,\sigma} : \quad \mathbb{R} \to (-\infty, 1], \quad t \mapsto \begin{cases} 0 & \text{if } t \leqslant \sigma - \frac{1}{M} \\ 1 - M \cdot |t - \sigma| & \text{if } t \geqslant \sigma - \frac{1}{M}, \end{cases} \tag{4}$$

and furthermore

$$\begin{aligned} \Delta^{(s)}_{M,y} : \quad & \mathbb{R}^d \to (-\infty, 1], \quad x \mapsto \left(\sum_{i=1}^{s} \Lambda_{M,y_i}(x_i) \right) - (s-1), \\ \vartheta^{(s)}_{M,y} : \quad & \mathbb{R}^d \to [0,1], \qquad x \mapsto \varrho(\Delta^{(s)}_{M,y}(x)), \end{aligned}$$

where, as before, $\varrho : \mathbb{R} \to \mathbb{R}$, $x \mapsto \max\{0, x\}$, denotes the ReLU activation function. A plot of $\Lambda_{M,\sigma}$ is shown in Figure 4.

With these definitions, the function $\vartheta^{(s)}_{M,y}$ satisfies the following properties:

Lemma A.1. *For $d \in \mathbb{N}$, $s \in [d]$, $M \geqslant 1$, $y \in [0,1]^d$, and $p \in (0, \infty]$, we have*

$$\operatorname{supp} \vartheta^{(s)}_{M,y} \subset y + (M^{-1} \cdot [-1,1]^s \times \mathbb{R}^{d-s})$$

and

$$\frac{1}{2} \cdot (2s)^{-s/p} \cdot M^{-s/p} \leqslant \|\vartheta^{(s)}_{M,y}\|_{L^p([0,1]^d)} \leqslant 2^{s/p} \cdot M^{-s/p}.$$

Proof. Let us first give a quick overview of the proof. The statement on the support of $\vartheta^{(s)}_{M,y}$ follows by observing that $\Delta^{(s)}_{M,y}(x) > 0$ can only happen if $\Lambda_{M,y_i}(x_i) > 0$ for all $i \in [s]$. As $0 \leqslant \vartheta^{(s)}_{M,y} \leqslant 1$, the upper bound on the $L^p([0,1]^d)$ norm can then be estimated by the Lebesgue measure of the intersection of the support of $\vartheta^{(s)}_{M,y}$ and the hypercube $[0,1]^d$. For the lower bound we compute the measure of the intersection with a subset of the support on which it holds that $\vartheta^{(s)}_{M,y} \geqslant \frac{1}{2}$.

We start by proving the statement on the support of $\vartheta^{(s)}_{M,y}$. If $0 \neq \vartheta^{(s)}_{M,y}(x)$, then $\Delta^{(s)}_{M,y}(x) > 0$, meaning $\sum_{i=1}^{s} \Lambda_{M,y_i}(x_i) > s - 1$. Because of $\Lambda_{M,y_i}(x_i) \in (-\infty, 1]$ for all $i \in [s]$, this is only possible if $\Lambda_{M,y_i}(x_i) > 0$ for all $i \in [s]$. Directly from the definition of Λ_{M,y_i} (see also Figure 4), this implies $|x_i - y_i| \leqslant \frac{1}{M}$ for all $i \in [s]$, meaning $x \in y + (M^{-1}[-1,1]^s \times \mathbb{R}^{d-s})$. This proves the first claim.

Regarding the second claim, define $y^* := (y_1, \ldots, y_s) \in \mathbb{R}^s$, and, for $k \in \mathbb{N}$, denote by λ_k the Lebesgue measure on $\mathbb{R}^k$. Then, since $[0,1]^d \cap \operatorname{supp} \vartheta^{(s)}_{M,y} \subset (y^* + M^{-1}[-1,1]^s) \times [0,1]^{d-s}$ and $0 \leqslant \vartheta^{(s)}_{M,y} \leqslant 1$, we see that

$$\|\vartheta^{(s)}_{M,y}\|_{L^p([0,1]^d)} \leqslant \left(\lambda_s\left(y^* + M^{-1}\,[-1,1]^s\right)\right)^{1/p} = \left(\frac{2}{M}\right)^{s/p} = 2^{s/p} M^{-s/p}.$$

For the converse estimate, let us also write $x^* = (x_1, \dots, x_s)$ for $x \in \mathbb{R}^d$. Then, if $x \in \mathbb{R}^d$ satisfies $x^* \in y^* + \frac{1}{2Ms}[-1,1]^s$, we see

$$y_i - \frac{1}{M} \leqslant y_i - \frac{1}{2Ms} \leqslant x_i \leqslant y_i + \frac{1}{2Ms} \quad \text{for } i \in [s].$$

By definition of Λ_{M,y_i}, this implies $\Lambda_{M,y_i}(x_i) = 1 - M \cdot |x_i - y_i| \geqslant 1 - \frac{1}{2s}$ and hence

$$\Delta_{M,y}^{(s)}(x) = \left(\sum_{i=1}^{s} \Lambda_{M,y_i}(x_i)\right) - (s-1) \geqslant s - \frac{1}{2} - (s-1) = \frac{1}{2},$$

so that $\vartheta_{M,y}^{(s)}(x) \geqslant \frac{1}{2}$.

Finally, it is not difficult to show, that

$$\lambda_d\big(\big\{x \in [0,1]^d \colon x^* \in y^* + \tfrac{1}{2Ms}[-1,1]^s\big\}\big) = \lambda_s\big([0,1]^s \cap (y^* + \tfrac{1}{2Ms}[-1,1]^s)\big) \geqslant (2Ms)^{-s},$$

see Grohs & Voigtlaender (2021, Equation (A.1)) for the details. Overall, we thus see

$$\|\vartheta_{M,y}^{(s)}\|_{L^p([0,1]^d)} \geqslant \frac{1}{2} \cdot (2Ms)^{-s/p}. \qquad \square$$

Note that a compactly supported (non-trivial) function such as $\vartheta_{M,y}^{(s)}$ can only be represented by ReLU networks with more than two layers, see Blum & Li (1991, Section 3). For this reason, we focus on the case $L \in \mathbb{N}_{\geqslant 3}$ in this paper. Next, we show that scaled versions of the hat functions $\vartheta_{M,y}^{(s)}$ can be represented using neural networks of a suitable architecture and with a suitable bound on the magnitude of the coefficients. We begin with the (more interesting) case where the exponent q that determines the regularization of the weights satisfies $q \geqslant 2$.

Lemma A.2. *Let $d \in \mathbb{N}$, $L \in \mathbb{N}_{\geqslant 3}$, $B \in \mathbb{N}_{\geqslant 3}$, $c > 0$, $q \in [2, \infty]$, and $s \in \mathbb{N}$ with $s \leqslant \min\{\frac{B}{3}, d\}$. Then, there exists a constant*

$$\lambda \geqslant c^L \cdot B^{(L-1)(1-\frac{2}{q})}/12$$

such that

$$\nu \cdot \frac{\lambda}{Ms} \cdot \vartheta_{M,y}^{(s)} \in \mathcal{H}^q_{(d,B,\dots,B,1),c} \qquad \forall\, M \in \mathbb{N}, \nu \in \{\pm 1\}, \text{ and } y \in [0,1]^d,$$

where the B in $(d, B, \dots, B, 1)$ appears $L-1$ times.

Proof. Let $M \in \mathbb{N}$, $y \in [0,1]^d$, and $\nu \in \{\pm 1\}$ be fixed. We will now construct the coefficients $((W^1, b^1), \dots, (W^L, b^L))$ of a neural network with the following properties:

1. The first two layers $((W^1, b^1), (W^2, b^2))$ output at any of their B output dimensions the function $C_1 \cdot \Lambda_{M,y}^{(s)}$ for a suitable scaling factor $C_1 = C_1(c, M, s, B, q) > 0$.

2. The following activation function yields $C_1 \cdot \vartheta_{M,y}^{(s)} = \varrho\big(C_1 \cdot \Lambda_{M,y}^{(s)}\big)$ for all output dimensions.

3. Each of the layers $((W^3, b^3), \dots, (W^{L-1}, b^{L-1}))$ scales the previous output by another factor $C_2 = C_2(c, B, q) > 0$, leading to the output $C_1 C_2^{L-3} \cdot \vartheta_{M,y}^{(s)}$ in any of the B output dimensions. This construction uses the fact that all intermediate outputs are positive by construction such that the intermediate ReLU activation functions ϱ just act as identities.

4. The last layer (W^L, b^L) now computes the sum of the previous outputs scaled by another factor $C_3 = C_3(c, B, q) > 0$ and multiplied by ν, such that the final one-dimensional output equals $\nu B C_1 C_2^{L-3} C_3 \cdot \vartheta_{M,y}^{(s)}$. The result follows by setting $\lambda = B C_1 C_2^{L-3} C_3 M s$ and choosing the scaling factors C_1, C_2, and C_3 as large as possible, constrained by the width B and the regularization given by c and q.

Define $r := \lfloor B/(3s) \rfloor$, noting that $r \geqslant 1$, since $s \leqslant B/3$. We first introduce a few notations: We write $0_{k\times n}$ for the $k \times n$ matrix with all entries being zero; similarly, we write $1_{k\times n}$ for the $k \times n$ matrix with all entries being one. Furthermore, we denote by $(e_1, \ldots, e_d)$ the standard basis of $\mathbb{R}^d$, and define

$$\begin{aligned} I_s &:= (e_1 \mid \cdots \mid e_s) \in \mathbb{R}^{d\times s}, \\ \alpha &:= \left(\tfrac{M^{-1}-y_1}{2} \middle| \tfrac{M^{-1}-y_2}{2} \middle| \cdots \middle| \tfrac{M^{-1}-y_s}{2}\right) \in \mathbb{R}^{1\times s}, \\ \beta &:= \big(-y_1 \mid -y_2 \mid \cdots \mid -y_s\big) \in \mathbb{R}^{1\times s}, \\ \gamma &:= \left(\tfrac{s-1}{s}\tfrac{1}{2M} \middle| \cdots \middle| \tfrac{s-1}{s}\tfrac{1}{2M}\right) = \frac{s-1}{s}\frac{1}{2M} \cdot 1_{1\times s} \in \mathbb{R}^{1\times s}. \end{aligned} \tag{5}$$

We note that all entries of these matrices and vectors are elements of $[-1, 1]$. Using these matrices and vectors, we now define

$$W^1 := \frac{c}{(3sr)^{1/q}} \Big(\underbrace{I_s/2|I_s|0_{d\times s}| \cdots |I_s/2|I_s|0_{d\times s}}_{r \text{ blocks of } (I_s/2|I_s|0_{d\times s})} |0_{d\times(B-3rs)} \Big)^T \in \mathbb{R}^{B\times d},$$

$$b^1 := \frac{c}{(3sr)^{1/q}} \Big(\underbrace{\alpha \mid \beta \mid \gamma \mid \cdots \mid \alpha \mid \beta \mid \gamma}_{r \text{ blocks of } (\alpha|\beta|\gamma)} \mid 0 \mid \cdots \mid 0 \Big)^T \in \mathbb{R}^B,$$

and furthermore

$$W^2 := \frac{c}{(3srB)^{1/q}} \Big(\underbrace{1_{B\times s} \mid -1_{B\times s} \mid -1_{B\times s} \mid \cdots \mid 1_{B\times s} \mid -1_{B\times s} \mid -1_{B\times s}}_{r \text{ blocks of } (1_{B\times s}|-1_{B\times s}|-1_{B\times s})} \mid 0_{B\times(B-3rs)} \Big) \in \mathbb{R}^{B\times B},$$

$$b^2 := (0 \mid \cdots \mid 0)^T \in \mathbb{R}^B,$$

where we note that $B - 3rs \geqslant 0$ since $r = \lfloor B/(3s) \rfloor$. It is straightforward to verify that $\|W^1\|_{\ell^q}, \|W^2\|_{\ell^q}, \|b^1\|_{\ell^q}, \|b^2\|_{\ell^q} \leqslant c$. Furthermore, we define

$$W^i := \frac{c}{B^{2/q}} 1_{B\times B} \quad \text{and} \quad b^i := (0|\cdots|0)^T \in \mathbb{R}^B \qquad \text{for } 3 \leqslant i \leqslant L-1,$$

and finally $W^L := \frac{\nu\cdot c}{B^{1/q}}(1|\cdots|1) \in \mathbb{R}^{1\times B}$ and $b^L := (0) \in \mathbb{R}^1$. Again, it is straightforward to verify that $\|W^i\|_{\ell^q}, \|b^i\|_{\ell^q} \leqslant c$ for $3 \leqslant i \leqslant L-1$ and also that $\|W^L\|_{\ell^q}, \|b^L\|_{\ell^q} \leqslant c$. Therefore, setting $\Phi := ((W^1, b^1), \ldots, (W^L, b^L))$, we have $R(\Phi) \in \mathcal{H}^q_{(d,B,\ldots,B,1),c}$; it thus remains to verify that $R(\Phi) = \nu \cdot \frac{\lambda}{Ms} \cdot \vartheta^{(s)}_{M,y}$ for a constant λ as in the statement of the lemma.

To see this, we note for any $x \in \mathbb{R}^d$ and $j \in [d]$ that

$$\begin{aligned} \varrho\big(\tfrac{x_j}{2} + \tfrac{M^{-1}-y_j}{2}\big) - \varrho(x_j - y_j) &= \tfrac{1}{2}\varrho\big(x_j - y_j + M^{-1}\big) - \varrho(x_j - y_j) \\ &= \begin{cases} 0 & \text{if } x_j \leqslant y_j - M^{-1} \\ \frac{1}{2M} \cdot (1 - M \cdot |x_j - y_j|) & \text{if } y_j - M^{-1} < x_j \leqslant y_j \\ \frac{1}{2M} \cdot (1 - M \cdot |x_j - y_j|) & \text{if } x_j > y_j \end{cases} \\ &= \tfrac{1}{2M}\Lambda_{M,y_j}(x_j). \end{aligned} \tag{6}$$

For notational convenience we further define $\phi^i(x) := \varrho(W^i x + b^i)$ for $i \in [L]$. Then, we observe for $x \in \mathbb{R}^B$ and $i \in [B]$ that

$$[\phi^2(x)]_i = \frac{c}{(3rsB)^{1/q}} \sum_{b=0}^{r-1} \sum_{j=1}^{s} \Big(x_{3sb+j} - x_{3sb+s+j} - x_{3sb+2s+j} \Big).$$

Therefore, we see for arbitrary $x \in \mathbb{R}^d$ and $i \in [B]$ that

$$\begin{aligned} \big[\big(\phi^2 \circ \varrho \circ \phi^1\big)(x)\big]_i &= \frac{c^2}{(3rs)^{2/q}B^{1/q}} \sum_{b=0}^{r-1} \sum_{j=1}^{s} \Big(\varrho\big(\tfrac{x_j}{2} + \tfrac{M^{-1}-y_j}{2}\big) - \varrho(x_j - y_j) - \varrho\big(\tfrac{s-1}{s}\tfrac{1}{2M}\big) \Big) \\ &= \frac{c^2}{2M(3rs)^{2/q}B^{1/q}} \sum_{b=0}^{r-1} \sum_{j=1}^{s} \Big(\Lambda_{M,y_j}(x_j) - \tfrac{s-1}{s} \Big) \\ &= \frac{c^2 r}{2M(3rs)^{2/q}B^{1/q}} \Delta^{(s)}_{M,y}(x). \end{aligned}$$

Hence, it holds that

$$\big(\varrho \circ \phi^2 \circ \varrho \circ \phi^1\big)(x) = \frac{c^2 r}{2M(3rs)^{2/q}B^{1/q}} \cdot \vartheta^{(s)}_{M,y}(x) \cdot (1|\cdots|1)^T \in \mathbb{R}^B.$$

Next, for $3 \leqslant i \leqslant L-1$, we see for arbitrary $\kappa \geqslant 0$ and $j \in [B]$ that

$$\big[\big(\varrho \circ \phi^i\big)\big(\kappa \cdot (1|\cdots|1)^T\big)\big]_j = \varrho\big(\textstyle\sum_{\ell=1}^B [W^i]_{j,\ell}\, \kappa\big) = \varrho(cB^{1-\frac{2}{q}}\kappa) = cB^{1-\frac{2}{q}}\kappa,$$

meaning

$$\big(\varrho \circ \phi^i\big)\big(\kappa(1 \mid \cdots \mid 1)^T\big) = cB^{1-\frac{2}{q}}\kappa \cdot (1 \mid \cdots \mid 1)^T.$$

Therefore, we conclude

$$\big(\varrho \circ \phi^{L-1} \circ \varrho \circ \phi^{L-2} \circ \cdots \circ \varrho \circ \phi^1\big)(x) = \frac{c^{L-1}\, r\, (B^{1-\frac{2}{q}})^{L-3}}{2M(3rs)^{2/q}B^{1/q}} \vartheta^{(s)}_{M,y}(x) \cdot (1 \mid \cdots \mid 1)^T \in \mathbb{R}^B.$$

All in all, this easily implies

$$R(\Phi)(x) = \frac{\nu}{B^{1/q}} \sum_{i=1}^{B} \frac{c^L r\, (B^{1-\frac{2}{q}})^{L-3}}{2M(3rs)^{2/q}B^{1/q}} \vartheta^{(s)}_{M,y}(x) = \nu \cdot \frac{c^L\, (B^{1-\frac{2}{q}})^{L-2}(3rs)^{1-\frac{2}{q}}}{6Ms} \vartheta^{(s)}_{M,y}(x).$$

It therefore remains to recall that $r = \lfloor B/(3s) \rfloor \geqslant 1$, so that $2r \geqslant 1 + r > \frac{B}{3s}$ and hence $3rs \geqslant \frac{B}{2}$. Since also $1 - \frac{2}{q} \geqslant 0$, this implies $(3rs)^{1-\frac{2}{q}} \geqslant (B/2)^{1-\frac{2}{q}} \geqslant B^{1-\frac{q}{2}}/2$, which finally shows

$$\lambda := \frac{c^L\, (B^{1-\frac{2}{q}})^{L-2}(3rs)^{1-\frac{2}{q}}}{6} \geqslant \frac{c^L \cdot B^{(L-1)(1-\frac{2}{q})}}{12}. \qquad \square$$

Now, we also consider the case $q \leqslant 2$. We remark that in the case $q = 2$, the next lemma only agrees with Lemma A.2 up to a constant factor. This is a proof artifact and is inconsequential for the questions we are interested in.

Lemma A.3. *Let $d \in \mathbb{N}$, $L \in \mathbb{N}_{\geqslant 3}$, $B \in \mathbb{N}_{\geqslant 3}$, $c > 0$, $q \in [1,2]$, and $s \in \mathbb{N}$ with $s \leqslant \min\{d, \frac{B}{3}\}$. Then, we have*

$$\nu \cdot \frac{c^L s^{1-\frac{2}{q}}/(2 \cdot 3^{2/q})}{Ms} \vartheta^{(s)}_{M,y} \in \mathcal{H}^q_{(d,B,\ldots,B,1),c} \quad \forall\, M \in \mathbb{N}, \nu \in \{\pm 1\},\ \textit{and}\ y \in [0,1]^d,$$

where the B in $(d, B, \ldots, B, 1)$ appears $L-1$ times.

Proof. The proof idea is similar to the one of Lemma A.2. However, we only realize a scaled version of the function $\vartheta^{(s)}_{M,y}$ in the *first* coordinate of the outputs after the first two layers. As in the proof of Lemma A.2, we denote by $(e_1, \ldots, e_d)$ the standard basis of $\mathbb{R}^d$, and we write $0_{k\times n}$ and $1_{k\times n}$ for the $k \times n$ matrices which have all entries equal to zero or one, respectively. Moreover, we use the matrices and vectors $I_s, \alpha, \beta, \gamma$ defined in Equation (5). With this setup, define

$$W^1 := \frac{c}{(3s)^{1/q}} \cdot \big(I_s/2 \,\big|\, I_s \,\big|\, 0_{d\times(B-2s)}\big)^T \in \mathbb{R}^{B\times d},$$
$$b^1 := \frac{c}{(3s)^{1/q}} \cdot \big(\alpha \,\big|\, \beta \,\big|\, \gamma \,\big|\, 0_{1\times(B-3s)}\big)^T \in \mathbb{R}^B.$$

Note that these definitions make sense since $2s \leqslant 3s \leqslant B$. Further, define $b^2 := (0|\cdots|0)^T \in \mathbb{R}^B$ and

$$W^2 := \frac{c}{(3s)^{1/q}} \begin{pmatrix} 1_{1\times s} & -1_{1\times 2s} & 0_{1\times(B-3s)} \\ 0_{(B-1)\times s} & 0_{(B-1)\times 2s} & 0_{(B-1)\times(B-3s)} \end{pmatrix} \in \mathbb{R}^{B\times B}.$$

Next, for $3 \leqslant i \leqslant L-1$, define $b^i := (0|\cdots|0)^T \in \mathbb{R}^B$ and

$$W^i := c \cdot \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{B\times B},$$

and finally let $W^L := \nu \cdot c \cdot (1|0|\cdots|0) \in \mathbb{R}^{1\times B}$ and $b^L := (0) \in \mathbb{R}^1$. It is straightforward to verify that $\|W^j\|_{\ell^q} \leqslant c$ and $\|b^j\|_{\ell^q} \leqslant c$ for all $1 \leqslant j \leqslant L$. Therefore, $R(\Phi) \in \mathcal{H}^q_{(d,B,\dots,B,1),c}$ for $\Phi := ((W^1,b^1),\dots,(W^L,b^L))$. It therefore remains to show that $R(\Phi) = \nu \cdot \frac{c^L s^{1-\frac{2}{q}}/(2\cdot 3^{2/q})}{Ms}\vartheta^{(s)}_{M,y}$.

For notational convenience we define $\phi^i(x) := \varrho(W^i x + b^i)$ for $i \in [L]$. Then we note for $3 \leqslant i \leqslant L-1$ that $\big(\varrho\circ\phi^i\big)(x) = \big(c\cdot\varrho(x_1)\,|\,0\,|\,\cdots\,|\,0\big)^T$. This easily implies

$$\big(\varrho\circ\phi^{L-1}\circ\varrho\circ\phi^{L-2}\circ\cdots\circ\varrho\circ\phi^3\big)(x) = \big(c^{L-3}\cdot\varrho(x_1)\,|\,0\,|\,\cdots\,|\,0\big)^T,$$

and therefore

$$\big(\phi^L\circ\varrho\circ\phi^{L-1}\circ\cdots\circ\varrho\circ\phi^3\big)(x) = \nu\cdot c^{L-2}\,\varrho(x_1) \quad \text{for } x\in\mathbb{R}^B.$$

Finally, an application of Equation (6) shows that

$$\begin{aligned}
[(\varrho\circ\phi^2\circ\varrho\circ\phi^1)(x)]_1 &= \frac{c^2}{(3s)^{2/q}}\varrho\bigg(\sum_{i=1}^s\bigg(\varrho\big(\tfrac{x_i}{2}+\tfrac{M^{-1}-y_i}{2}\big)-\varrho(x_i-y_i)-\varrho\big(\tfrac{s-1}{s}\tfrac{1}{2M}\big)\bigg)\bigg)\\
&= \frac{c^2}{2M\cdot(3s)^{2/q}}\varrho\bigg(\bigg(\sum_{i=1}^s\Lambda_{M,y_i}(x_i)\bigg)-(s-1)\bigg)\\
&= \frac{c^2}{2M\cdot(3s)^{2/q}}\varrho(\Delta^{(s)}_{M,y}(x)) = \frac{c^2}{2M\cdot(3s)^{2/q}}\vartheta^{(s)}_{M,y}(x).
\end{aligned}$$

Overall, we thus see as claimed that

$$R(\Phi)(x) = \nu\cdot c^{L-2}\cdot\frac{c^2}{2M\cdot(3s)^{2/q}}\cdot\vartheta^{(s)}_{M,y}(x) = \nu\cdot\frac{c^L s^{1-\frac{2}{q}}/(2\cdot 3^{2/q})}{Ms}\cdot\vartheta^{(s)}_{M,y}(x). \qquad \square$$

Remark A.4. *A straightforward adaptation of the proof shows that the same statement holds for $\mathcal{H}^q_{(d,B,N_2,\dots,N_{L-1},1),c}$ instead of $\mathcal{H}^q_{(d,B,\dots,B,1),c}$, for arbitrary $N_2,\dots,N_{L-1}\in\mathbb{N}$.*

A.2 A GENERAL LOWER BOUND

We now show that any target class containing a large number of (shifted) hat functions has a large optimal error.

Theorem A.5. *Let $d, m \in \mathbb{N}$, $s \in [d]$, and $M := 8\lceil m^{1/s}\rceil$. Assume that $U \subset C([0,1]^d)$ satisfies*

$$u_0 + \nu\cdot\frac{\lambda}{Ms}\vartheta^{(s)}_{M,y} \in U \qquad \forall\,\nu\in\{\pm1\} \text{ and } y\in[0,1]^d$$

for certain $\lambda > 0$ and $u_0 \in C([0,1]^d)$. Then,

$$\mathrm{err}^{MC}_m(U, L^p([0,1]^d)) \geqslant \frac{\lambda/4}{(32s)^{1+\frac{s}{p}}}\cdot m^{-\frac{1}{p}-\frac{1}{s}} \qquad \forall\, p\in[1,\infty].$$

The general idea of the proof is sketched in Section 2.3. In what follows we provide the technical details.

Proof. The proof is divided into five steps.

Step 1: Define $k := \lceil m^{1/s}\rceil$ and let $y^\ell := \frac{(1,\dots,1)}{8k} + \frac{\ell-(1,\dots,1)}{4k} \in [0,1]^d$ for $\ell\in[4k]^d$. Furthermore, let $\Gamma := [4k]^s\times\{(1,\dots,1)\} \subset [4k]^d$. With

$$f_{\ell,\nu} := u_0 + \nu\cdot\frac{\lambda}{Ms}\vartheta^{(s)}_{M,y^\ell} \quad \text{for} \quad (\ell,\nu)\in\Gamma\times\{\pm1\}, \tag{7}$$

it holds by assumption that

$$f_{\ell,\nu}\in U \quad \forall\,(\ell,\nu)\in\Gamma\times\{\pm1\}. \tag{8}$$

Furthermore, since $M = 8k$, Lemma A.1 and a moment's thought reveal that

$$\forall\,(\ell,\nu),(\ell',\nu')\in\Gamma\times\{\pm1\}:\ \ell\neq\ell' \Rightarrow \mathrm{supp}(f_{\ell,\nu}-u_0)^\circ\cap\mathrm{supp}(f_{\ell',\nu'}-u_0)^\circ = \varnothing, \tag{9}$$

where we note that $\operatorname{supp}(f_{\ell,\nu} - u_0) = \operatorname{supp} \vartheta^{(s)}_{M,y^\ell}$.

Step 2: Let[5] $A \in \operatorname{Alg}_{2m}(U, L^p)$ be arbitrary and $\mathbf{x} = \mathbf{x}(u_0) = (x_1, \dots, x_{2m}) \in \left([0,1]^d\right)^{2m}$ as described before Equation (1). Put

$$I_{\mathbf{x}} := \left\{ \ell \in \Gamma : \, \forall\, i \in [2m] : \, \vartheta^{(s)}_{M,y^\ell}(x_i) = 0 \right\}.$$

We now show that

$$|I_{\mathbf{x}}| \geqslant (4k)^s - 2m. \tag{10}$$

To see this we will estimate the cardinality of the complement set $I^c_{\mathbf{x}} := \Gamma \backslash I_{\mathbf{x}}$ from above. For $\ell \in I^c_{\mathbf{x}}$ there must exist $i_\ell \in [2m]$ with $\vartheta^{(s)}_{M,y^\ell}(x_{i_\ell}) \neq 0$ and hence $x_{i_\ell} \in \operatorname{supp}\left(\vartheta^{(s)}_{M,y^\ell}\right)^{\circ}$. The map $I^c_{\mathbf{x}} \to [2m]$, $\ell \mapsto i_\ell$, is thus injective due to (9). Therefore $|I^c_{\mathbf{x}}| \leqslant 2m$ and thus $|I_{\mathbf{x}}| \geqslant |\Gamma| - 2m$, which is (10). Furthermore, the definition of $I_{\mathbf{x}}$, combined with the definition of $f_{\ell,\nu}$ in (7) and the condition that A can only depend on the samples $\mathbf{x}$ and the values of the input function at these samples, directly imply that

$$\forall\, (\ell, \nu) \in \Gamma \times \{\pm 1\} : \, \ell \in I_{\mathbf{x}} \Rightarrow A(f_{\ell,\nu}) = A(u_0). \tag{11}$$

Step 3: Recalling our notation for the average in Section 1.2, it holds that

$$\begin{aligned}
\sum_{\substack{\ell \in \Gamma \\ \nu \in \{\pm 1\}}} \|f_{\ell,\nu} - A(f_{\ell,\nu})\|_{L^p} &= \frac{1}{(4k)^s} \sum_{\ell \in \Gamma} \left(\frac{1}{2} \|f_{\ell,-1} - A(f_{\ell,-1})\|_{L^p} + \frac{1}{2} \|f_{\ell,1} - A(f_{\ell,1})\|_{L^p} \right) \\
&\geqslant \frac{1}{(4k)^s} \sum_{\ell \in I_{\mathbf{x}}} \left(\frac{1}{2} \|f_{\ell,-1} - A(f_{\ell,-1})\|_{L^p} + \frac{1}{2} \|f_{\ell,1} - A(f_{\ell,1})\|_{L^p} \right) && (12) \\
&= \frac{|I_{\mathbf{x}}|}{(4k)^s} \sum_{\ell \in I_{\mathbf{x}}} \left(\frac{1}{2} \|f_{\ell,-1} - A(f_{\ell,-1})\|_{L^p} + \frac{1}{2} \|f_{\ell,1} - A(f_{\ell,1})\|_{L^p} \right) \\
&\geqslant \frac{1}{2} \sum_{\ell \in I_{\mathbf{x}}} \left(\frac{1}{2} \|f_{\ell,-1} - A(f_{\ell,-1})\|_{L^p} + \frac{1}{2} \|f_{\ell,1} - A(f_{\ell,1})\|_{L^p} \right) && (13) \\
&= \frac{1}{2} \sum_{\ell \in I_{\mathbf{x}}} \left(\frac{1}{2} \|f_{\ell,-1} - A(u_0)\|_{L^p} + \frac{1}{2} \|f_{\ell,1} - A(u_0)\|_{L^p} \right) && (14) \\
&\geqslant \frac{1}{2} \sum_{\ell \in I_{\mathbf{x}}} \left\| \frac{\lambda}{Ms} \vartheta^{(s)}_{M,y^\ell} \right\|_{L^p} && (15) \\
&\geqslant \frac{1}{2} \cdot (2s)^{-\frac{s}{p}-1} \cdot \lambda \cdot M^{-\frac{s}{p}-1} && (16) \\
&\geqslant \frac{1}{2} \cdot (2s)^{-\frac{s}{p}-1} \cdot \lambda \cdot 16^{-\frac{s}{p}-1} \cdot m^{-\frac{1}{p}-\frac{1}{s}} && (17) \\
&= \frac{\lambda}{2 \cdot (32s)^{\frac{s}{p}+1}} \cdot m^{-\frac{1}{p}-\frac{1}{s}}.
\end{aligned}$$

Here, (12) follows since $I_{\mathbf{x}} \subset \Gamma$; (13) follows from $k = \lceil m^{\frac{1}{s}} \rceil$ and (10); (14) follows from (11); (15) follows from the triangle inequality and (7); (16) follows from Lemma A.1; and (17) follows from the definition of M, which implies that $M \leqslant 8\, m^{1/s} + 8 \leqslant 16\, m^{1/s}$.

Step 4: Let $(\mathbf{A}, \mathbf{m}) \in \operatorname{Alg}^{MC}_m(U, L^p)$ be arbitrary with $\mathbf{A} = (A_\omega)_{\omega \in \Omega}$ for a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Put $\Omega_0 := \{\omega \in \Omega : \, \mathbf{m}(\omega) \leqslant 2m\}$. Since the Markov inequality implies that

$$m \geqslant \mathbb{E}[\mathbf{m}] \geqslant 2m \cdot \mathbb{P}(\Omega_0^c),$$

it follows that

$$\mathbb{P}(\Omega_0) \geqslant \frac{1}{2}. \tag{18}$$

[5]For notational convenience, we abbreviate $L^p([0,1]^d)$ by L^p in this proof.

Step 5: We finally estimate for $(\mathbf{A}, \mathbf{m})$ as in Step 4 that

$$\sup_{u\in U} \mathbb{E}\left[\|u - A_\omega(u)\|_{L^p}\right] \geqslant \sum_{\substack{\ell\in\Gamma \\ \nu\in\{\pm 1\}}} \mathbb{E}\left[\|f_{\ell,\nu} - A_\omega(f_{\ell,\nu})\|_{L^p}\right] \tag{19}$$

$$\geqslant \mathbb{E}\left[\mathbb{1}_{\Omega_0}(\omega) \sum_{\substack{\ell\in\Gamma \\ \nu\in\{\pm 1\}}} \|f_{\ell,\nu} - A_\omega(f_{\ell,\nu})\|_{L^p}\right]$$

$$\geqslant \mathbb{P}(\Omega_0) \cdot \frac{\lambda}{2\cdot(32s)^{\frac{s}{p}+1}} \cdot m^{-\frac{1}{p}-\frac{1}{s}} \tag{20}$$

$$\geqslant \frac{\lambda/4}{(32s)^{\frac{s}{p}+1}} \cdot m^{-\frac{1}{p}-\frac{1}{s}}. \tag{21}$$

Here, (19) follows from (8); (20) follows from Step 3 (note that $A_\omega \in \mathrm{Alg}_{2m}(U, L^p)$ for $\omega \in \Omega_0$); and (21) follows from (18).

Since $(\mathbf{A}, \mathbf{m}) \in \mathrm{Alg}_m^{MC}(U, L^p)$ was arbitrary, this implies the desired statement. □

Remark A.6. *Close inspection of the proof of Theorem 2.2 shows that one can replace the point samples $u(x_i)$ by $Tu(x_i)$, where $T : U \to C([0,1]^d)$ is any local operator*[6]*. Since any differential operator is a local operator, our lower bounds also hold if we measure point samples of a differential operator applied to u, as it is commonly done in the context of so-called physics-informed neural networks (Raissi et al., 2019).*

B Proof of the upper bound in Section 2.4

We first provide an auxiliary result which bounds the spectral norm $\|W\|_{\ell^2\to\ell^2}$ of a matrix W by its entry-wise ℓ^q norm.

Lemma B.1. *Let $W \in \mathbb{R}^{N\times M}$ and $q \in [1,\infty]$. Then it holds that*

$$\|W\|_{\ell^2\to\ell^2} \leqslant \begin{cases} \|W\|_{\ell^q} & \text{if } q \leqslant 2 \\ (\sqrt{NM})^{1-\frac{2}{q}} \cdot \|W\|_{\ell^q} & \text{if } q \geqslant 2. \end{cases}$$

Proof. We first note that $\|W\|_{\ell^2} = \|W\|_F$, the Frobenius norm of the matrix W. It is well-known that the Frobenius norm satisfies $\|W\|_{\ell^2\to\ell^2} \leqslant \|W\|_F$. Since we could not locate a convenient reference, we reproduce the elementary proof: The Cauchy-Schwarz inequality implies that

$$\|Wx\|_{\ell^2}^2 = \sum_{i=1}^N \left|\sum_{j=1}^M W_{i,j}x_j\right|^2 \leqslant \sum_{i=1}^N \left(\sum_{j=1}^M |W_{i,j}|^2 \sum_{j=1}^M |x_j|^2\right) = \|W\|_{\ell^2}^2 \cdot \|x\|_{\ell^2}^2,$$

which implies the claim. Thus, we see for $q \leqslant 2$ that $\|W\|_{\ell^2\to\ell^2} \leqslant \|W\|_{\ell^2} \leqslant \|W\|_{\ell^q}$. Clearly, the same estimate holds for complex-valued matrices and vectors as well.

Now, to handle the case $q \geqslant 2$, we first note for $q = \infty$ and $W \in \mathbb{C}^{N\times M}$ and $x \in \mathbb{C}^M$ that

$$\|Wx\|_{\ell^2}^2 = \sum_{i=1}^N \left|\sum_{j=1}^M W_{i,j}\, x_j\right|^2 \leqslant \sum_{i=1}^N \left(\sum_{j=1}^M |W_{i,j}|^2 \sum_{j=1}^M |x_j|^2\right) \leqslant \|x\|_{\ell^2}^2 \cdot \|W\|_{\ell^\infty}^2 \cdot NM.$$

This proves the claim in case of $q = \infty$. Finally, for $q \in (2,\infty)$, we choose $\theta = \frac{2}{q}$, so that $\frac{1}{q} = \frac{\theta}{2} + \frac{1-\theta}{\infty}$. Thus, applying the *Riesz-Thorin interpolation theorem* (see, e.g., Folland, 1999, Theorem 6.27) to the linear map $(\mathbb{C}^{N\times M}, \|\cdot\|_{\ell^q}) \to (\mathbb{C}^N, \|\cdot\|_{\ell^2})$, $W \mapsto Wx$, shows for each $x \in \mathbb{C}^M$ that

$$\|Wx\|_{\ell^2} \leqslant (\sqrt{NM})^{1-\theta} \cdot \|W\|_{\ell^q} = (\sqrt{NM})^{1-\frac{2}{q}} \cdot \|W\|_{\ell^q},$$

which completes the proof[7]. □

[6] This means that if $f = g$ on a neighborhood of $x \in [0,1]^d$, then $(Tf)(x) = (Tg)(x)$.

[7] We consider complex matrices and vectors, since the Riesz-Thorin theorem applies as stated only for the complex setting.

Next, let us define the Lipschitz constant $\mathrm{Lip}_{\ell^q}(\phi)$ of a function $\phi : \mathbb{R}^d \to \mathbb{R}^k$ with respect to the ℓ^2 norm by

$$\mathrm{Lip}_{\ell^q}(\phi) := \sup_{x,y\in\mathbb{R}^d, x\neq y} \frac{\|\phi(x)-\phi(y)\|_{\ell^q}}{\|x-y\|_{\ell^q}}.$$

Note that the Lipschitz constant of an affine-linear mapping $x \mapsto Wx + b$ equals the spectral norm $\|W\|_{\ell^2\to\ell^2}$. Thus, we can use the previous lemma to bound the Lipschitz constant of neural network realizations $R(\Phi) \in \mathcal{H}^q_{(N_0,\dots,N_L),c}$ in terms of their architecture $(N_0, \dots, N_L)$ and the regularization on their weights (given by $\max_{1\leqslant i\leqslant L} \max\{\|W^i\|_{\ell^q}, \|b^i\|_{\ell^q}\} \leqslant c$).

Lemma B.2. *Let $L \in \mathbb{N}$, $q \in [1,\infty]$, $c > 0$, and $N_0, \dots, N_L \in \mathbb{N}$. Then, each $R(\Phi) \in \mathcal{H}^q_{(N_0,\dots,N_L),c}$ satisfies*

$$\mathrm{Lip}_{\ell^2}(R(\Phi)) \leqslant \begin{cases} c^L & \text{if } q \leqslant 2 \\ c^L \cdot (\sqrt{N_0 N_L}\cdot N_1 \cdots N_{L-1})^{1-2/q} & \text{if } q \geqslant 2. \end{cases}$$

Proof. Let $R(\Phi) \in \mathcal{H}^q_{(N_0,\dots,N_L),c}$ be arbitrary. By definition, this means

$$R(\Phi) = \phi^L \circ \varrho \circ \phi^{L-1} \circ \cdots \circ \varrho \circ \phi^1,$$

where ϱ acts componentwise, and where the affine-linear maps $\phi^i : \mathbb{R}^{N_{i-1}} \to \mathbb{R}^{N_i}$ are of the form $\phi^i(x) = W^i x + b^i$, with $W^i \in \mathbb{R}^{N_i \times N_{i-1}}$ and $\|W^i\|_{\ell^q} \leqslant c$.

The ReLU activation function $\varrho : \mathbb{R} \to \mathbb{R}$, $x \mapsto \max\{0, x\}$, is easily seen to satisfy $|\varrho(x) - \varrho(y)| \leqslant |x - y|$ for $x, y \in \mathbb{R}$. This implies that

$$\mathrm{Lip}_{\ell^2}(R(\Phi)) = \mathrm{Lip}_{\ell^2}(\phi^L \circ \varrho \circ \phi^{L-1} \circ \cdots \circ \varrho \circ \phi^1) \leqslant \prod_{i=1}^{L} \mathrm{Lip}_{\ell^2}(\phi^i). \tag{22}$$

Lemma B.1 establishes for $i \in [L]$ that

$$\mathrm{Lip}_{\ell^2}(\phi^i) \leqslant \begin{cases} c & \text{if } q \leqslant 2 \\ c \cdot (\sqrt{N_{i-1}N_i})^{1-\frac{2}{q}} & \text{if } q \geqslant 2, \end{cases}$$

which, together with (22), proves the claim. □

Note that we can estimate the error of reconstructing Lipschitz continuous functions from samples by piecewise constant interpolation. Together with Lemma B.2, this allows us to construct a (non-adaptive, deterministic) algorithm for reconstructing neural networks from samples.

Lemma B.3. *Let $d \in \mathbb{N}$. Then, for every $m \in \mathbb{N}$, there exist points $x_1, \dots, x_m \in [0,1]^d$ and a map $\Theta_m : \mathbb{R}^m \to L^\infty([0,1]^d)$ satisfying*

$$\big\|\Theta_m\big(u(x_1), \dots, u(x_m)\big) - u\big\|_{L^\infty([0,1]^d)} \leqslant \mathrm{Lip}_{\ell^2}(u) \cdot 2\sqrt{d} \cdot m^{-1/d} \tag{23}$$

for every function $u : [0,1]^d \to \mathbb{R}$ with $\mathrm{Lip}_{\ell^2}(u) < \infty$.

Proof. Let $m \in \mathbb{N}$ be arbitrary and choose $K := \lfloor m^{1/d} \rfloor \geqslant 1$. Write

$$\{x_1, \dots, x_{K^d}\} = \tfrac{(1,\dots,1)}{2K} + \big\{\tfrac{0}{K}, \tfrac{1}{K}, \dots, \tfrac{K-1}{K}\big\}^d \quad \text{noting that} \quad [0,1]^d = \bigcup_{i=1}^{K^d} x_i + [-\tfrac{1}{2K}, \tfrac{1}{2K}]^d.$$

Hence, choosing $Q_i := (x_i + [-\frac{1}{2K}, \frac{1}{2K}]^d) \setminus \bigcup_{j=1}^{i-1}(x_j + [-\frac{1}{2K}, \frac{1}{2K}]^d)$, we get $[0,1]^d = \biguplus_{i=1}^{K^d} Q_i$, where the union is disjoint.

Note that $K^d \leqslant m$ and choose arbitrary points $x_{K^d+1}, \dots, x_m \in [0,1]^d$. Furthermore, define

$$\Theta_m : \quad \mathbb{R}^m \to L^\infty([0,1]^d), \quad (a_1, \dots, a_m) \mapsto \sum_{i=1}^{K^d} a_i \cdot \mathbb{1}_{Q_i}.$$

To prove Equation (23), let $u : [0,1]^d \to \mathbb{R}$ be arbitrary with $\mathrm{Lip}_{\ell^2}(u) < \infty$. For arbitrary $x \in [0,1]^d$, there then exists a unique $i \in [K^d]$ satisfying $x \in Q_i \subset x_i + [-\frac{1}{2K}, \frac{1}{2K}]^d$, and in particular $\|x - x_i\|_{\ell^2} \leqslant \sqrt{d}/(2K)$. Therefore,

$$\begin{aligned}\big|\Theta_m\big(u(x_1),\dots,u(x_m)\big)(x) - u(x)\big| &= |u(x_i) - u(x)| \\ &\leqslant \mathrm{Lip}_{\ell^2}(u) \cdot \|x_i - x\|_{\ell^2} \leqslant \mathrm{Lip}_{\ell^2}(u) \cdot \frac{\sqrt{d}}{2K}.\end{aligned}$$

Since $x \in [0,1]^d$ was arbitrary, this implies

$$\big\|\Theta_m(u(x_1),\dots,u(x_m)) - u\big\|_{L^\infty([0,1]^d)} \leqslant \mathrm{Lip}_{\ell^2}(u) \cdot \frac{\sqrt{d}}{2K}.$$

Finally, we note that $K = \lfloor m^{1/d} \rfloor \geqslant 1$ implies $2K \geqslant 1 + K > m^{1/d}$, which proves the claim. □

Note that the proof above requires to convert a Lipschitz constant with respect to the ℓ_2 norm to an ℓ_∞ estimate which costs a factor $\sqrt{d}$ and contributes to the gap between our lower and upper bound.

Remark B.4. *Note that our upper and lower bounds in Theorems 1.1 and 1.4 are asymptotically sharp with respect to the number of samples m, the regularization parameter c, and the network depth L but not fully sharp with respect to the multiplicative factor depending on d and q only. Given m many samples, a combination of Theorems 1.1 and 1.4 shows that the optimal achievable L^∞ reconstruction error ε for reconstructing neural networks with L layers up to width $3d$ and coefficients bounded by c in the ℓ^q norm satisfies*

$$\left\{\begin{array}{ll} \frac{1}{256\cdot 3^{2/q}} \cdot c^L \cdot d^{-\frac{2}{q}} \cdot m^{-\frac{1}{d}} & \\ \frac{1}{1536\cdot d} \cdot c^L \cdot (3d)^{(L-1)(1-\frac{2}{q})} \cdot m^{-\frac{1}{d}} & \end{array}\right\} \leqslant \varepsilon \leqslant \left\{\begin{array}{ll} \sqrt{d} \cdot c^L \cdot m^{-\frac{1}{d}} & \text{if } q \leqslant 2 \\ d^{1-\frac{1}{q}} \cdot c^L \cdot (3d)^{(L-1)(1-\frac{2}{q})} \cdot m^{-\frac{1}{d}} & \text{if } q > 2. \end{array}\right\}$$

For moderate input dimensions d the upper and lower bounds are quite tight, but for larger d there remains a gap. However, in that case the lower bound for m is already intractable (at least if $\varepsilon \ll 1/d$ or if $c \gg 1$ and L is large) so that the upper bound is merely of academic interest.

C HYPERPARAMETERS USED IN THE NUMERICAL EXPERIMENTS

Table 1: General hyperparameters for the experiments in Figure 1 and Section 3.

Description	Value	Variable
Experiment		
precision	float64	
GPUs per training	1 (NVIDIA GTX-1080, RTX-2080Ti, A40, or A100)	
Deep learning algorithms		
optimizer	Adam	
initialization of coefficients (W^ℓ, b^ℓ)	$\mathcal{U}([-\sqrt{1/N_{\ell-1}}, \sqrt{1/N_{\ell-1}}])$	
activation function	ReLU	ϱ
learning rate scheduler	exponential decay	
initial / final learning rate	10^{-4} / 10^{-6}	
decay frequency	every epoch	
Evaluation		
number of samples	2^{24}	J
distribution of samples	$\mathcal{U}([-0.5, 0.5]^d)$	
evaluation norm	$\{1, 2, \infty\}$	p
number of evaluations	5 (evenly spaced over all epochs)	

Table 2: Hyperparameters specific to the experiment in Figure 1.

Description	Value	Variable
Experiment		
samples	10^3	m
dimension	1	d
Target function		
sinusoidal function	$x \mapsto \log(\sin(50x) + 2) + \sin(5x)$	u
Deep learning algorithm		
depth of architecture	22	L
width of architecture	50	$N_1, \ldots, N_{L-1}$
batch-size	20	
number of epochs	5000	

Table 3: Hyperparameters specific to the experiments in Section 3.

Description	Value	Variable
Experiment		
samples	$\{10^2, 10^3, 10^4, 10^5\}$	m
dimension	$\{1, 3\}$	d
Target functions		
number of teachers	40	$\lvert\widehat{U}\rvert$
depth of teacher architecture	5	L
width of teacher architecture	32	$N_1, \ldots, N_{L-1}$
activation of teacher	ReLU	ϱ
teacher coefficient norm	∞	q
teacher coefficient norm bound	0.5	c
distribution of coefficients	$\mathcal{U}([-0.5, 0.5])$	
Deep learning algorithms		
number of seeds	3	$\lvert\widehat{\Omega}\rvert$
depth of student architecture	5	L
width of student architecture	$\{32, 512, 2048\}$	$N_1, \ldots, N_{L-1}$
batch-size	$\{100, m/5, m/50\}$	
number of epochs	500 (if batch-size $= 100$), 5000 (else)	

Appendix

Julius Berner

Bridging theory and application in deep learning

mail@jberner.info • jberner.info • 0000-0002-5648-648X • juliusberner • julius-berner

Education

PhD Student
2018–present

University of Vienna — *Vienna, Austria*

- Research group: Mathematical Data Science group led by Philipp Grohs
- Thesis: *Mathematical analysis of deep learning with applications to Kolmogorov equations*

Master of Science in Mathematics
2016–2018

University of Vienna — *Vienna, Austria*

- Area of specialisation: Applied Mathematics and Scientific Computing
- Pass with distinction; performance scholarship awarded for outstanding academic achievement
- Thesis: *Solving stochastic differential equations and Kolmogorov equations by means of deep learning and Multilevel Monte Carlo simulation*

Bachelor of Science in Mathematics
2012–2016

University of Vienna — *Vienna, Austria*

- Pass with distinction; performance scholarship awarded for outstanding academic achievement
- Theses: *Tarski–Seidenberg theorem*; *Discrete cosine transform in image processing*

Employment & Internships

Research Assistant
08/2018–present

University of Vienna — *Vienna, Austria*

- Supervisor: Philipp Grohs
- Funding: Austrian Science Fund Project Position

AI Research Intern (FAIR Labs)
05/2022–08/2022

Meta — *New York, USA*

- Supervisors: Karen Ullrich and Matthew Muckley
- Topic: Generative modeling and variational inference in the context of neural compression

Research Intern (AI Automation)
05/2020–10/2020

NVIDIA — *Remote*

- Supervisors: Christoph Angerer and Elmar Haussmann
- Topic: Semi-supervised & active learning for object detection in autonomous driving

Selected Publications

- J. **Berner**, P. Grohs, and F. Voigtlaender, "Learning ReLU networks to high uniform accuracy is intractable," in *International Conference on Learning Representations*, 2023.
- J. **Berner**, L. Richter, and K. Ullrich, "An optimal control perspective on diffusion-based generative modeling," in *NeurIPS 2022 Workshop on Score-Based Methods*, 2022.
- L. Richter and J. **Berner**, "Robust SDE-based variational formulations for solving linear PDEs via deep learning," in *Proceedings of the 39th International Conference on Machine Learning*, vol. 162 of *Proceedings of Machine Learning Research*, pp. 18649–18666, PMLR, 2022.
- J. **Berner**, P. Grohs, G. Kutyniok, and P. Petersen, *The Modern Mathematics of Deep Learning*, p. 1–111. Cambridge University Press, 2022.
- C. M. Verdun, T. Fuchs, P. Harar, D. Elbrächter, D. S. Fischer, J. **Berner**, P. Grohs, F. J. Theis, and F. Krahmer, "Group testing for SARS-CoV-2 allows for up to 10-fold efficiency increase across realistic scenarios and testing strategies," *Frontiers in Public Health*, vol. 9, p. 1205, 2021.
- J. **Berner**, M. Dablander, and P. Grohs, "Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 16615–16627, Curran Associates, Inc., 2020.
- J. **Berner**, P. Grohs, and A. Jentzen, "Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black–Scholes partial differential equations," *SIAM Journal on Mathematics of Data Science*, vol. 2, no. 3, pp. 631–657, 2020.
- J. **Berner**, D. Elbrächter, and P. Grohs, "How degenerate is the parametrization of neural networks with the ReLU activation function?," in *Advances in Neural Information Processing Systems*, vol. 32, pp. 7790–7801, Curran Associates, Inc., 2019.
- J. **Berner**, D. Elbrächter, P. Grohs, and A. Jentzen, "Towards a regularity theory for ReLU networks–chain rule and global error estimates," in *2019 13th International conference on Sampling Theory and Applications (SampTA)*, pp. 1–5, IEEE, 2019.

Talks & Poster Presentations

12/2022	**NeurIPS 2022 Workshop on Score-Based Methods** An optimal control perspective on diffusion-based generative modeling	New Orleans Convention Center *New Orleans, USA*
07/2022	**39th International Conference on Machine Learning (ICML)** Robust SDE-Based Variational Formulations for Solving Linear PDEs via Deep Learning	Baltimore Convention Center *Baltimore, USA*
12/2020	**34th Annual Conference on Neural Information Processing Systems (NeurIPS)** Numerically Solving Parametric Families of High-Dimensional Kolmogorov Partial Differential Equations via Deep Learning	*Virtual-only*
12/2019	**33rd Annual Conference on Neural Information Processing Systems (NeurIPS)** How degenerate is the parametrization of neural networks with the ReLU activation function?	Vancouver Convention Center *Vancouver, Canada*
10/2019	**School on Mathematical and Computational Aspects of Machine Learning** Regularity theory of deep ReLU networks in the context of partial differential equations	Scuola Normale Superiore *Pisa, Italy*
09/2019	**Oberwolfach Workshop: Innovative Approaches to the Numerical Approximation of PDEs** Empirical risk minimization over deep neural networks overcomes the curse of dimensionality in the numerical approximation of Kolmogorov equations	Oberwolfach Research Institute for Mathematics *Oberwolfach, Germany*
07/2019	**13th International Conference on Sampling Theory and Applications (SampTA)** Towards a regularity theory for ReLU networks–chain rule and global error estimates	Université de Bordeaux *Bordeaux, France*
06/2019	**RTG Summer Lectures** Mathematical Analysis of Deep Learning Based Methods for Solving High-Dimensional PDEs	University of Chicago *Chicago, USA*
06/2019	**NYU CDS Research Group Meeting** Mathematical Analysis of Deep Learning Based Methods for Solving High-Dimensional PDEs	NYU Center for Data Science *New York, USA*
02/2019	**90st GAMM Annual Meeting** Analysis of the Generalization Error: Empirical Risk Minimization Over Deep Neural Networks Overcomes the Curse of Dimensionality in the Numerical Approximation of Black-Scholes PDEs	Technical University of Vienna *Vienna, Austria*

Teaching

02/2020	**Machine Learning in Physics: VDSP-ESI Winter School** Support-vector machines in scikit-learn, TensorFlow, and PyTorch; reproducible scientific ML experiments; transfer learning using fast.ai; solving Kolmogorov PDEs by means of deep learning	Erwin Schrödinger Institute *Vienna, Austria*
11/2019	**Oberwolfach Graduate Seminar: Mathematics of Deep Learning** Introduction to TensorFlow; solving Kolmogorov PDEs by means of deep learning; neural network approximation in TensorFlow; neural network training in the overparametrized setting	Mathematical Research and Conference Center *Będlewo, Poland*
02/2015–07/2015	**Numerical Analysis Exercise Class** Teaching assistant for the numerical analysis exercises and associate implementations in Matlab	University of Vienna *Vienna, Austria*

Community Service

02/2019–02/2020	**Seminar Organizer** Organizer of the Deep Learning Seminar at the Faculty of Mathematics	University of Vienna *Vienna, Austria*
03/2019–present	**Reviewer** Reviews for ICML 2023, ICLR 2023, ICML 2022 (outstanding reviewer), ICLR 2022 (highlighted reviewer), NeurIPS 2022, NeurIPS 2021, JUQ, ACHA, SISC, AAP, ISIT, SampTA	