Neural Samplers and PDE Solvers Based on Diffusion Processes

Julius Berner

Caltech

May 14, 2024

Overview

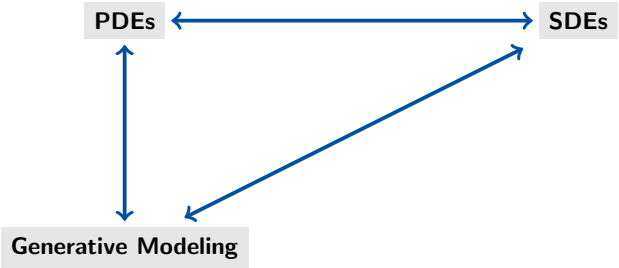
Overview

- SDE-based losses for neural PDE solver

PDEs ⟷ **SDEs**

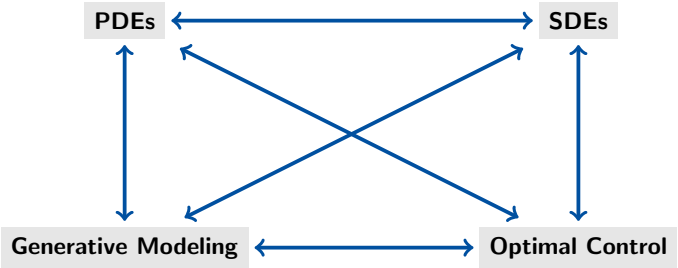
Overview

- SDE-based losses for neural PDE solver
- The role of PDEs in generative modeling and sampling



Overview

- SDE-based losses for neural PDE solver
- The role of PDEs in generative modeling and sampling
- Diffusion models, Schrödinger bridges, and optimal control



Neural PDE solver

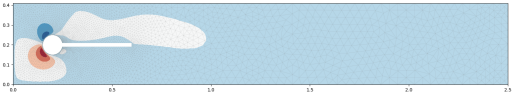
Partial differential equation (PDEs)

Differential equations [...] represent the most powerful tool humanity has ever created for making sense of the material world. – Steven Strogatz (2009)

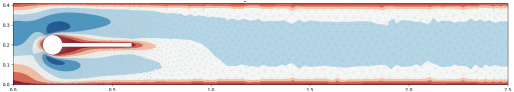
Partial differential equation (PDEs)

Differential equations [...] represent the most powerful tool humanity has ever created for making sense of the material world. – Steven Strogatz (2009)

Fluid dynamics

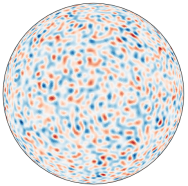


Pressure

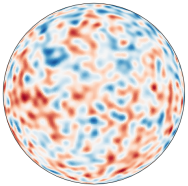


Horizontal velocity

Atmospheric and oceanic modeling



Initial geopotential height



After 1h

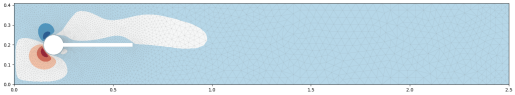
Fig. 1: M. A. Rahman, R. J. George, M. Elleithy, D. Leibovici, Z. Li, B. Bonev, C. White, J. Berner, et al. Pretraining codomain attention neural operators for solving multiphysics PDEs. arXiv preprint arXiv:2403.12553, 2024

Fig. 2: M. Liu-Schiaffini, J. Berner, B. Bonev, T. Kurth, K. Azizzadenesheli, and A. Anandkumar. Neural operators with localized integral and differential kernels. ICML, 2024

Partial differential equation (PDEs)

Differential equations [...] represent the most powerful tool humanity has ever created for making sense of the material world. – Steven Strogatz (2009)

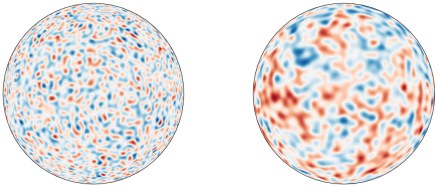
Fluid dynamics



Pressure

Horizontal velocity

Atmospheric and oceanic modeling



Initial geopotential height

After 1h

Goal: Approximate the solution $u \in V$ of a PDE

$$Lu = 0 \quad \text{on } D,$$

given by a differential operator L on a domain D with suitable boundary conditions on ∂D.

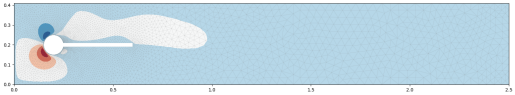
Fig. 1: M. A. Rahman, R. J. George, M. Elleithy, D. Leibovici, Z. Li, B. Bonev, C. White, J. Berner, et al. Pretraining codomain attention neural operators for solving multiphysics PDEs. arXiv preprint arXiv:2403.12553, 2024

Fig. 2: M. Liu-Schiaffini, J. Berner, B. Bonev, T. Kurth, K. Azizzadenesheli, and A. Anandkumar. Neural operators with localized integral and differential kernels. ICML, 2024

Partial differential equation (PDEs)

Differential equations [...] represent the most powerful tool humanity has ever created for making sense of the material world. – Steven Strogatz (2009)

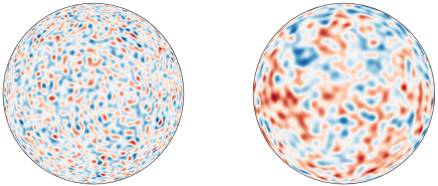
Fluid dynamics



Pressure

Horizontal velocity

Atmospheric and oceanic modeling



Initial geopotential height

After 1h

Goal: Approximate the solution $u \in V$ of a PDE

$$Lu = 0 \quad \text{on } D,$$

given by a differential operator L on a domain D with suitable boundary conditions on ∂D.

Diffusion models

- $D = \mathbb{R}^d \times [0, T]$
- $Lu = \partial_t u + \operatorname{div}(u\mu) - \frac{1}{2}\sigma^2 \Delta u$
- $u(\cdot, 0) = p_{\text{target}}$

Fig. 1: M. A. Rahman, R. J. George, M. Elleithy, D. Leibovici, Z. Li, B. Bonev, C. White, J. Berner, et al. Pretraining codomain attention neural operators for solving multiphysics PDEs. arXiv preprint arXiv:2403.12553, 2024

Fig. 2: M. Liu-Schiaffini, J. Berner, B. Bonev, T. Kurth, K. Azizzadenesheli, and A. Anandkumar. Neural operators with localized integral and differential kernels. ICML, 2024

Selected classical methods

Finite difference method (FDM)

Approximate the differential operator L by difference quotients on a grid.

Selected classical methods

Finite difference method (FDM)

Approximate the differential operator L by difference quotients on a grid.

For instance:

$$\partial_{x_i x_i} u(x) \approx \frac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2}$$

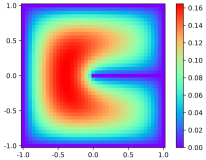


Fig. 1: W. E, J. Han, and A. Jentzen. Deep learning-based numerical methods for high-dimensional parabolic pdes and backward sdes. Communications in mathematics and statistics, 5(4):349–380, 2017

Selected classical methods

Finite difference method (FDM)

Approximate the differential operator L by difference quotients on a grid.

For instance:

$$\partial_{x_i x_i} u(x) \approx \frac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2}$$

$\Rightarrow$ linear system for

$$(u(x + jhe_i))_{i,j}$$

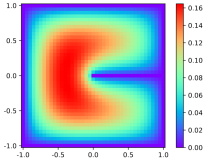


Fig. 1: W. E, J. Han, and A. Jentzen. Deep learning-based numerical methods for high-dimensional parabolic pdes and backward sdes. Communications in mathematics and statistics, 5(4):349–380, 2017

Selected classical methods

Finite difference method (FDM)

Approximate the differential operator L by difference quotients on a grid.

Finite element method (FEM)

Approximate the function space V by a finite-dimensional space.

For instance:

$$\partial_{x_i x_i} u(x) \approx \frac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2}$$

$\Rightarrow$ linear system for

$$(u(x + jhe_i))_{i,j}$$

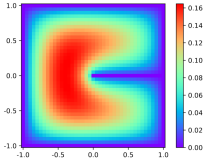


Fig. 1: W. E, J. Han, and A. Jentzen. Deep learning-based numerical methods for high-dimensional parabolic pdes and backward sdes. Communications in mathematics and statistics, 5(4):349–380, 2017

Selected classical methods

Finite difference method (FDM)

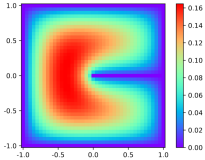
Approximate the differential operator L by difference quotients on a grid.

For instance:

$$\partial_{x_i x_i} u(x) \approx \frac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2}$$

$\Rightarrow$ linear system for

$$(u(x + jhe_i))_{i,j}$$



Finite element method (FEM)

Approximate the function space V by a finite-dimensional space.

For instance (Galerkin approx. for linear L):

$$u(x) \approx \sum_{i=1}^{n} c_i v_i(x),$$

where $(v_i)_{i=1}^{n} \subset V$ are are suitable functions.

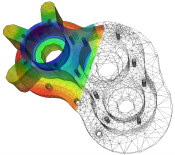


Fig. 2: https://de.wikipedia.org/wiki/Datei:Elmer-pump-heatequation.png

Fig. 1: W. E, J. Han, and A. Jentzen. Deep learning-based numerical methods for high-dimensional parabolic pdes and backward sdes. Communications in mathematics and statistics, 5(4):349–380, 2017

Selected classical methods

Finite difference method (FDM)

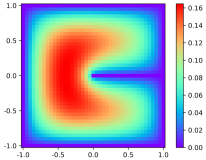
Approximate the differential operator L by difference quotients on a grid.

For instance:

$$\partial_{x_i x_i} u(x) \approx \frac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2}$$

$\Rightarrow$ linear system for

$$(u(x + jhe_i))_{i,j}$$



Finite element method (FEM)

Approximate the function space V by a finite-dimensional space.

For instance (Galerkin approx. for linear L):

$$u(x) \approx \sum_{i=1}^{n} c_i v_i(x),$$

where $(v_i)_{i=1}^n \subset V$ are are suitable functions.

$\Rightarrow$ linear system:

$$\sum_{i=1}^{n} c_i \langle L(v_i), v_j \rangle = \langle f, v_j \rangle$$

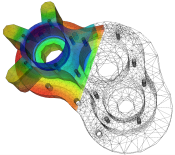


Fig. 2: https://de.wikipedia.org/wiki/Datei:Elmer-pump-heatequation.png

Fig. 1: W. E, J. Han, and A. Jentzen. Deep learning-based numerical methods for high-dimensional parabolic pdes and backward sdes. Communications in mathematics and statistics, 5(4):349–380, 2017

Selected classical methods

Finite difference method (FDM)

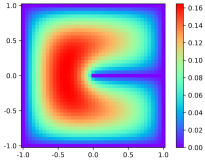
Approximate the differential operator L by difference quotients on a grid.

For instance:

$$\partial_{x_i x_i} u(x) \approx \frac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2}$$

$\Rightarrow$ linear system for

$$(u(x + jhe_i))_{i,j}$$



Finite element method (FEM)

Approximate the function space V by a finite-dimensional space.

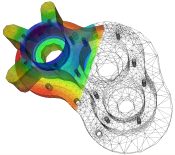
For instance (Galerkin approx. for linear L):

$$u(x) \approx \sum_{i=1}^{n} c_i v_i(x),$$

where $(v_i)_{i=1}^n \subset V$ are are suitable functions.

$\Rightarrow$ linear system:

$$\sum_{i=1}^{n} c_i \langle L(v_i), v_j \rangle = \langle f, v_j \rangle$$



👍 high accuracies, theoretical guarantees

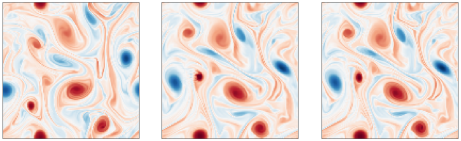
👎 needs to be tailored to the PDE, slow/infeasible for fine grids or high dims.

Fig. 2: https://de.wikipedia.org/wiki/Datei:Elmer-pump-heatequation.png

Fig. 1: W. E, J. Han, and A. Jentzen. Deep learning-based numerical methods for high-dimensional parabolic pdes and backward sdes. Communications in mathematics and statistics, 5(4):349–380, 2017

Neural PDE Solvers

Surrogate models / supervised:
Given data (from classical solvers, simulations, or real-world observations) learn the mapping from parameters/inputs to the PDE solution.

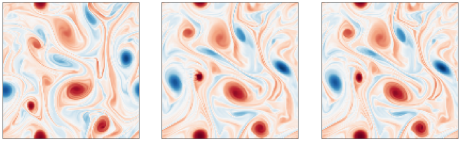


Initial Condition | Prediction (5 steps) | Ground Truth

Fig. 1: M. Liu-Schiaffini, J. Berner, B. Bonev, T. Kurth, K. Azizzadenesheli, and A. Anandkumar. Neural operators with localized integral and differential kernels. ICML, 2024

Neural PDE Solvers

Surrogate models / supervised:
Given data (from classical solvers, simulations, or real-world observations) learn the mapping from parameters/inputs to the PDE solution.



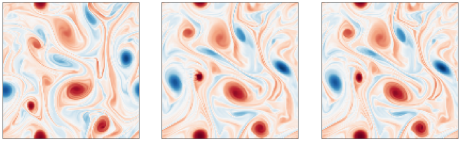
Initial Condition | Prediction (5 steps) | Ground Truth

👍 simple objective, PDE can be unknown
👎 needs (high-resolution) training data

Fig. 1: M. Liu-Schiaffini, J. Berner, B. Bonev, T. Kurth, K. Azizzadenesheli, and A. Anandkumar. Neural operators with localized integral and differential kernels. ICML, 2024

Neural PDE Solvers

Surrogate models / supervised:
Given data (from classical solvers, simulations, or real-world observations) learn the mapping from parameters/inputs to the PDE solution.



Initial Condition | Prediction (5 steps) | Ground Truth

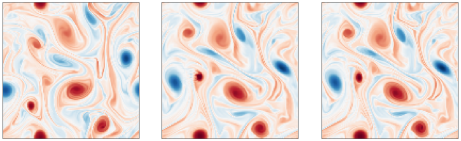
👍 simple objective, PDE can be unknown
👎 needs (high-resolution) training data

Variational approaches / unsupervised:
Minimize a variational formulation, where the minimizer corresponds to the PDE solution.

Fig. 1: M. Liu-Schiaffini, J. Berner, B. Bonev, T. Kurth, K. Azizzadenesheli, and A. Anandkumar. Neural operators with localized integral and differential kernels. ICML, 2024

Neural PDE Solvers

Surrogate models / supervised:
Given data (from classical solvers, simulations, or real-world observations) learn the mapping from parameters/inputs to the PDE solution.



Initial Condition | Prediction (5 steps) | Ground Truth

👍 simple objective, PDE can be unknown
👎 needs (high-resolution) training data

Variational approaches / unsupervised:
Minimize a variational formulation, where the minimizer corresponds to the PDE solution.

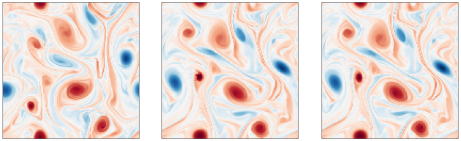
- PDE-residual (PINNs, Deep Galerkin)

$$\min_\theta \|Lu_\theta\| + \lambda \cdot \mathrm{BoundaryLoss}(u_\theta)$$

Fig. 1: M. Liu-Schiaffini, J. Berner, B. Bonev, T. Kurth, K. Azizzadenesheli, and A. Anandkumar. Neural operators with localized integral and differential kernels. ICML, 2024

Neural PDE Solvers

Surrogate models / supervised:
Given data (from classical solvers, simulations, or real-world observations) learn the mapping from parameters/inputs to the PDE solution.



Initial Condition Prediction (5 steps) Ground Truth

👍 simple objective, PDE can be unknown
👎 needs (high-resolution) training data

Variational approaches / unsupervised:
Minimize a variational formulation, where the minimizer corresponds to the PDE solution.

- PDE-residual (PINNs, Deep Galerkin)

$$\min_\theta \|Lu_\theta\| + \lambda \cdot \mathrm{BoundaryLoss}(u_\theta)$$

- Weak formulations / energy-based (Deep-Ritz, Friedrichs learning, ...)

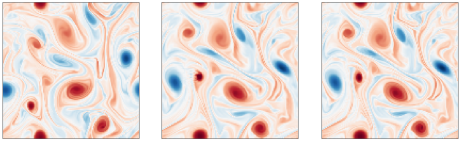
PINN Deep Ritz Ground Truth

Fig. 2: H. C. Nam, J. Berner, and A. Anandkumar. Solving Poisson equations using neural walk-on-spheres. In ICML, 2024

Fig. 1: M. Liu-Schiaffini, J. Berner, B. Bonev, T. Kurth, K. Azizzadenesheli, and A. Anandkumar. Neural operators with localized integral and differential kernels. ICML, 2024

Neural PDE Solvers

Surrogate models / supervised:
Given data (from classical solvers, simulations, or real-world observations) learn the mapping from parameters/inputs to the PDE solution.



Initial Condition | Prediction (5 steps) | Ground Truth

👍 simple objective, PDE can be unknown
👎 needs (high-resolution) training data

Variational approaches / unsupervised:
Minimize a variational formulation, where the minimizer corresponds to the PDE solution.

- PDE-residual (PINNs, Deep Galerkin)

$$\min_\theta \|Lu_\theta\| + \lambda \cdot \mathrm{BoundaryLoss}(u_\theta)$$

- Weak formulations / energy-based (Deep-Ritz, Friedrichs learning, ...)

PINN | Deep Ritz | Ground Truth

👍 applicable to most PDEs, unlimited data
👎 sensitive to hyperparameters, (higher-order) derivatives needed

Fig. 2: H. C. Nam, J. Berner, and A. Anandkumar. Solving Poisson equations using neural walk-on-spheres. In ICML, 2024

Fig. 1: M. Liu-Schiaffini, J. Berner, B. Bonev, T. Kurth, K. Azizzadenesheli, and A. Anandkumar. Neural operators with localized integral and differential kernels. ICML, 2024

SDE-based neural PDE solver

PDEs $\longleftrightarrow$ **SDEs**

Neural PDE solvers based on stochastic representations

Stochastic representations / (un)supervised:
For certain PDEs, solutions can be written as expectations over suitable random variables and we can regress noisy but cheap simulations.

Neural PDE solvers based on stochastic representations

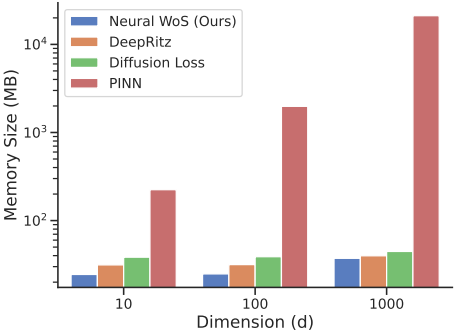
Stochastic representations / (un)supervised:
For certain PDEs, solutions can be written as expectations over suitable random variables and we can regress noisy but cheap simulations.

👍 without derivatives, include boundary conditions without penalty terms

Neural PDE solvers based on stochastic representations

Stochastic representations / (un)supervised:
For certain PDEs, solutions can be written as expectations over suitable random variables and we can regress noisy but cheap simulations.

👍 without derivatives, include boundary conditions without penalty terms



Figs.: H. C. Nam, J. Berner, and A. Anandkumar. Solving Poisson equations using neural walk-on-spheres. In ICML, 2024

Kolmogorov (backward) equations

$$\partial_t u = -\frac{1}{2}\,\mathrm{trace}\,\big(\underbrace{\sigma\sigma^\top}_{\text{diffusion coefficient}}\nabla^2 u\big)$$

Kolmogorov (backward) equations

$$\partial_t u = -\frac{1}{2}\,\mathrm{trace}\left(\underset{\uparrow}{\sigma\sigma^\top}\nabla^2 u\right) - \underset{\uparrow}{\mu}\cdot\nabla u$$

diffusion coefficient / drift coefficient

Kolmogorov (backward) equations

$$\partial_t u = -\frac{1}{2}\,\mathrm{trace}\left(\underset{\substack{\uparrow \\ \text{diffusion coefficient}}}{\sigma}\sigma^\top \nabla^2 u\right) - \underset{\substack{\uparrow \\ \text{drift coefficient}}}{\mu} \cdot \nabla u, \qquad u(x,T) = \underset{\substack{\uparrow \\ \text{terminal condition}}}{\varphi}(x).$$

Kolmogorov (backward) equations

$$\partial_t u = -\frac{1}{2}\,\mathrm{trace}\left(\underset{\substack{\uparrow \\ \text{diffusion coefficient}}}{\sigma\sigma^\top}\nabla^2 u\right) - \underset{\substack{\uparrow \\ \text{drift coefficient}}}{\mu}\cdot\nabla u, \qquad u(x,T) = \underset{\substack{\uparrow \\ \text{terminal condition}}}{\varphi}(x).$$

Method can be extended to models for drift–diffusion processes (e.g., diffusion models), optimal control problems (HJB equation), fluid dynamics, and option pricing.

Kolmogorov (backward) equations

$$\partial_t u = -\tfrac{1}{2}\,\mathrm{trace}\left(\underset{\substack{\uparrow \\ \text{diffusion coefficient}}}{\sigma\sigma^\top}\nabla^2 u\right) - \underset{\substack{\uparrow \\ \text{drift coefficient}}}{\mu}\cdot\nabla u, \qquad u(x,T) = \underset{\substack{\uparrow \\ \text{terminal condition}}}{\varphi(x)}.$$

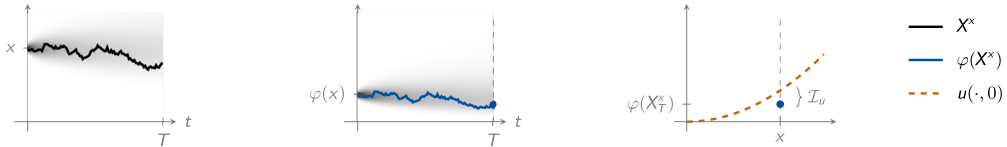
Method can be extended to models for drift–diffusion processes (e.g., diffusion models), optimal control problems (HJB equation), fluid dynamics, and option pricing.

Stochastic representation (Itô's lemma):

$$u(x,0) = \varphi(X_T^x) - \underbrace{\mathcal{I}_u}_{\text{noise with zero expectation}},$$

where X^x solves the **stochastic differential equation** (SDE)

$$\mathrm{d}X_t^x = \mu(X_t^x)\,\mathrm{d}t + \sigma(X_t^x)\,\mathrm{d}B_t, \quad X_0^x = x.$$



Kolmogorov (backward) equations

$$\partial_t u = -\tfrac{1}{2}\,\mathrm{trace}\left(\underset{\substack{\uparrow \\ \text{diffusion coefficient}}}{\sigma\sigma^\top}\nabla^2 u\right) - \underset{\substack{\uparrow \\ \text{drift coefficient}}}{\mu}\cdot\nabla u, \qquad u(x,T) = \underset{\substack{\uparrow \\ \text{terminal condition}}}{\varphi(x)}.$$

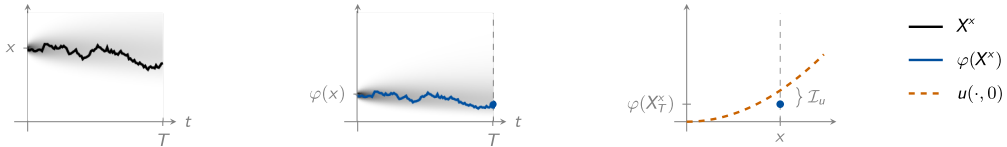
Method can be extended to models for drift–diffusion processes (e.g., diffusion models), optimal control problems (HJB equation), fluid dynamics, and option pricing.

Stochastic representation (Itô's lemma / Feynman-Kac formula):

$$u(x,0) = \varphi(X_T^x) - \underbrace{\mathcal{I}_u}_{\text{noise with zero expectation}} = \mathbb{E}\left[\varphi(X_T^x)\right]$$

where X^x solves the **stochastic differential equation** (SDE)

$$\mathrm{d}X_t^x = \mu(X_t^x)\,\mathrm{d}t + \sigma(X_t^x)\,\mathrm{d}B_t, \quad X_0^x = x.$$



Euler-Maruyama scheme and Monte Carlo sampling

Discretize the SDE

$$\mathrm{d}X_t^x = \mu(X_t^x)\,\mathrm{d}t + \sigma(X_t^x)\,\mathrm{d}B_t, \quad X_0^x = x,$$

using the Euler-Maruyama scheme with step size Δt:

$$\hat{X}_{t+\Delta t}^x = \hat{X}_t^x + \mu(\hat{X}_t^x)\Delta t + \sigma(\hat{X}_t^x)\underbrace{(B_{t+\Delta t} - B_t)}_{\sim\mathcal{N}(0,\Delta t)}.$$

P. E. Kloeden and E. Platen. Numerical Solution of Stochastic Differential Equations. Stochastic Modelling and Applied Probability. Springer, 1992

Euler-Maruyama scheme and Monte Carlo sampling

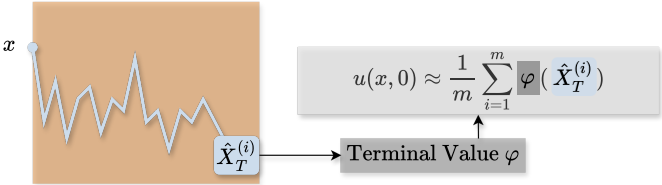
Discretize the SDE

$$\mathrm{d}X_t^x = \mu(X_t^x)\,\mathrm{d}t + \sigma(X_t^x)\,\mathrm{d}B_t, \quad X_0^x = x,$$

using the Euler-Maruyama scheme with step size Δt:

$$\hat{X}_{t+\Delta t}^x = \hat{X}_t^x + \mu(\hat{X}_t^x)\Delta t + \sigma(\hat{X}_t^x)\underbrace{(B_{t+\Delta t} - B_t)}_{\sim\mathcal{N}(0,\Delta t)}.$$

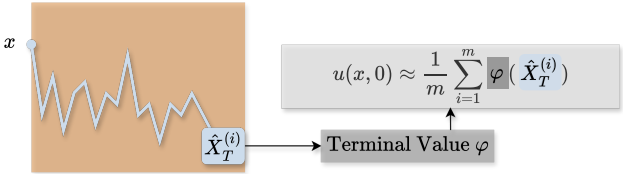
Use Monte Carlo (MC) sampling to approximate the solution $u(x,0) = \mathbb{E}\left[\varphi(X_T^x)\right]$ with MSE $\mathcal{O}(m^{-1} + \Delta t)$:



P. E. Kloeden and E. Platen. Numerical Solution of Stochastic Differential Equations. Stochastic Modelling and Applied Probability. Springer, 1992

SDE-based neural PDE solver

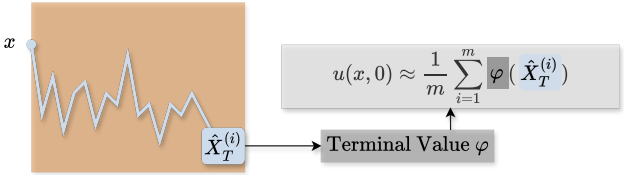
Monte Carlo sampling gives us **pointwise estimates** of $u(x, 0)$ for fixed $x \in \mathbb{R}^d$.



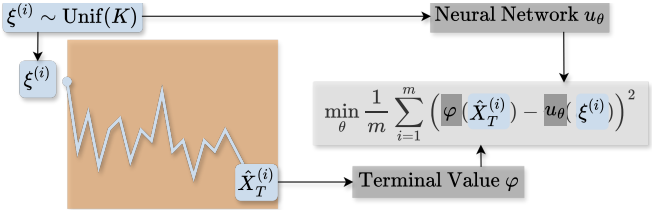
$$u(x, 0) \approx \frac{1}{m} \sum_{i=1}^{m} \varphi(\hat{X}_T^{(i)})$$

SDE-based neural PDE solver

Monte Carlo sampling gives us **pointwise estimates** of $u(x,0)$ for fixed $x \in \mathbb{R}^d$.



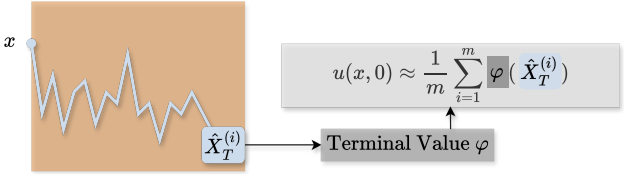
We can extend this framework to learn $u(\cdot,0)$ on a **domain** $K \subset \mathbb{R}^d$.



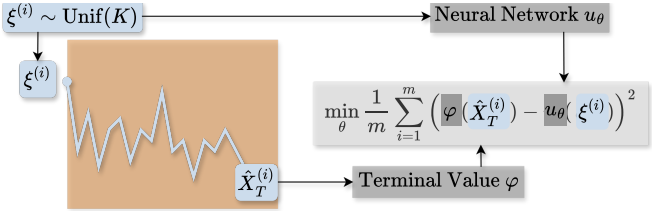
C. Beck, S. Becker, P. Grohs, N. Jaafari, and A. Jentzen. Solving the Kolmogorov PDE by means of deep learning. Journal of Scientific Computing, 88(3):1–28, 2021

SDE-based neural PDE solver

Monte Carlo sampling gives us **pointwise estimates** of $u(x,0)$ for fixed $x \in \mathbb{R}^d$.



We can extend this framework to learn $u(\cdot,0)$ on a **domain** $K \subset \mathbb{R}^d$.

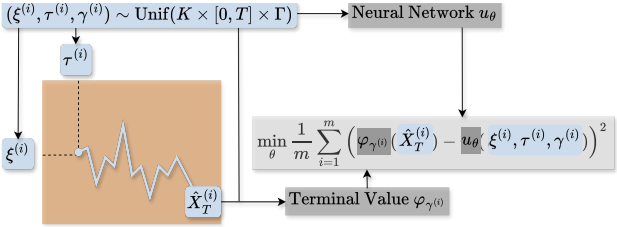


Variational formulation: $u(\cdot,0) = \mathrm{argmin}_v \, \mathbb{E}\left[(\varphi(X_T^\xi) - v(\xi))^2\right]$.

C. Beck, S. Becker, P. Grohs, N. Jaafari, and A. Jentzen. Solving the Kolmogorov PDE by means of deep learning. Journal of Scientific Computing, 88(3):1–28, 2021

SDE-based neural PDE solver

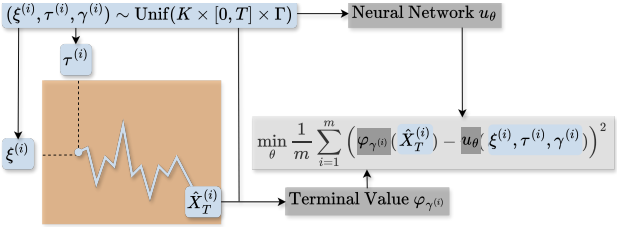
- We extend the framework to **parametric PDEs** and **space-time domains** $K \times [0, T]$.



J. **Berner**, M. Dablander, and P. Grohs. Numerically solving parametric families of high-dimensional Kolmogorov PDEs via deep learning. In NeurIPS, pages 16615–16627, 2020

SDE-based neural PDE solver

- We extend the framework to **parametric PDEs** and **space-time domains** $K \times [0, T]$.
- We prove bounds on the approx. and generalization error (can **overcome the curse of dimensionality**).



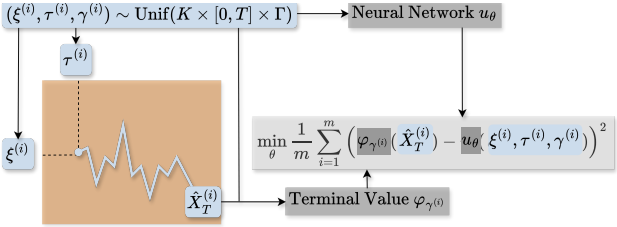
J. **Berner**, P. Grohs, and F. Voigtlaender. Learning ReLU networks to high uniform accuracy is intractable. In ICLR, 2023

J. **Berner**, P. Grohs, and A. Jentzen. Analysis of the generalization error [...]. SIAM Journal on Mathematics of Data Science, 2(3):631–657, 2020

J. **Berner**, M. Dablander, and P. Grohs. Numerically solving parametric families of high-dimensional Kolmogorov PDEs via deep learning. In NeurIPS, pages 16615–16627, 2020

SDE-based neural PDE solver

- We extend the framework to **parametric PDEs** and **space-time domains** $K \times [0, T]$.
- We prove bounds on the approx. and generalization error (can **overcome the curse of dimensionality**).
- We provide **variance-reduced losses** (based on approximating $\mathcal{I}_u$).



L. Richter and J. **Berner**. Robust SDE-based variational formulations for solving linear PDEs via deep learning. In ICML, pages 18649–18666, 2022

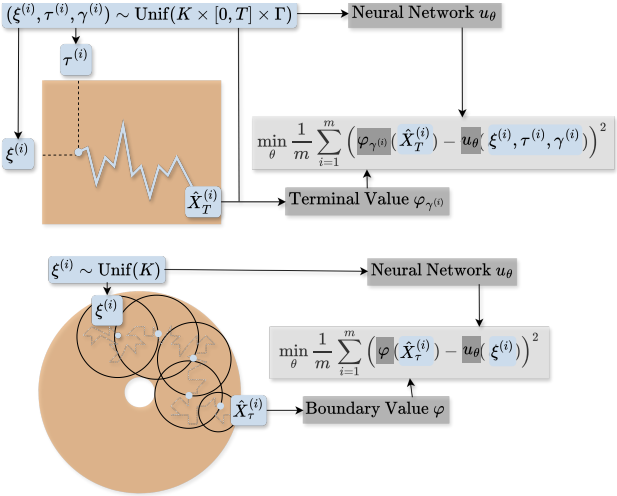
J. **Berner**, P. Grohs, and F. Voigtlaender. Learning ReLU networks to high uniform accuracy is intractable. In ICLR, 2023

J. **Berner**, P. Grohs, and A. Jentzen. Analysis of the generalization error [...]. SIAM Journal on Mathematics of Data Science, 2(3):631–657, 2020

J. **Berner**, M. Dablander, and P. Grohs. Numerically solving parametric families of high-dimensional Kolmogorov PDEs via deep learning. In NeurIPS, pages 16615–16627, 2020

SDE-based neural PDE solver

- We extend the framework to **parametric PDEs** and **space-time domains** $K \times [0, T]$.
- We prove bounds on the approx. and generalization error (can **overcome the curse of dimensionality**).
- We provide **variance-reduced losses** (based on approximating $\mathcal{I}_u$).
- We extend it to **boundary value problems** using walk-on-spheres.



H. C. Nam, J. Berner, and A. Anandkumar. Solving Poisson equations using neural walk-on-spheres. In ICML, 2024

L. Richter and J. **Berner**. Robust SDE-based variational formulations for solving linear PDEs via deep learning. In ICML, pages 18649–18666, 2022

J. **Berner**, P. Grohs, and F. Voigtlaender. Learning ReLU networks to high uniform accuracy is intractable. In ICLR, 2023

J. **Berner**, P. Grohs, and A. Jentzen. Analysis of the generalization error [...]. SIAM Journal on Mathematics of Data Science, 2(3):631–657, 2020

J. **Berner**, M. Dablander, and P. Grohs. Numerically solving parametric families of high-dimensional Kolmogorov PDEs via deep learning. In NeurIPS, pages 16615–16627, 2020

SDE-based neural PDE solver - Results

- We solve (parametric) PDEs from optimal control, chemistry, and finance with > 50 dims. (**out-of-scope for classical methods**).

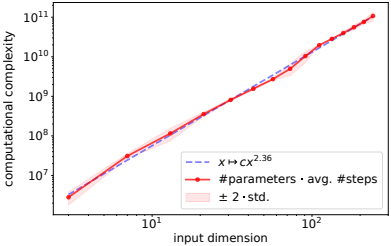
J. **Berner**, M. Dablander, and P. Grohs. Numerically solving parametric families of high-dimensional Kolmogorov PDEs via deep learning. In NeurIPS, pages 16615–16627, 2020

L. Richter and J. **Berner**. Robust SDE-based variational formulations for solving linear PDEs via deep learning. In ICML, pages 18649–18666, 2022

H. C. Nam, J. Berner, and A. Anandkumar. Solving Poisson equations using neural walk-on-spheres. In ICML, 2024

SDE-based neural PDE solver - Results

- We solve (parametric) PDEs from optimal control, chemistry, and finance with > 50 dims. (**out-of-scope for classical methods**).
- Numerical evidence that our method can **overcome the curse of dimensionality**.



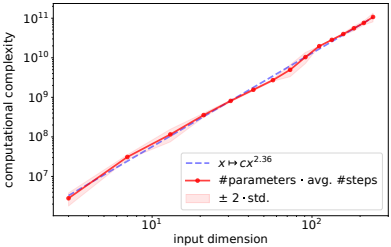
J. **Berner**, M. Dablander, and P. Grohs. Numerically solving parametric families of high-dimensional Kolmogorov PDEs via deep learning. In NeurIPS, pages 16615–16627, 2020

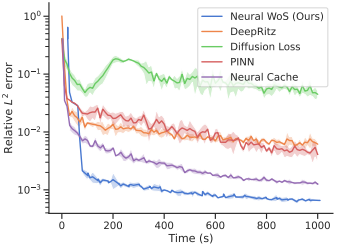
L. Richter and J. **Berner**. Robust SDE-based variational formulations for solving linear PDEs via deep learning. In ICML, pages 18649–18666, 2022

H. C. Nam, J. Berner, and A. Anandkumar. Solving Poisson equations using neural walk-on-spheres. In ICML, 2024

SDE-based neural PDE solver - Results

- We solve (parametric) PDEs from optimal control, chemistry, and finance with > 50 dims. (**out-of-scope for classical methods**).
- Numerical evidence that our method can **overcome the curse of dimensionality**.
- More **efficient and accurate** than competing methods (e.g., PINNs, Deep Ritz).





J. **Berner**, M. Dablander, and P. Grohs. Numerically solving parametric families of high-dimensional Kolmogorov PDEs via deep learning. In NeurIPS, pages 16615–16627, 2020

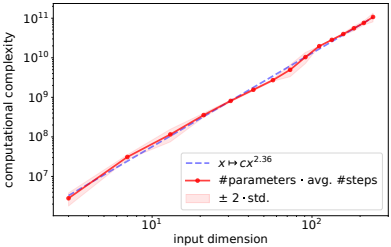
L. Richter and J. **Berner**. Robust SDE-based variational formulations for solving linear PDEs via deep learning. In ICML, pages 18649–18666, 2022

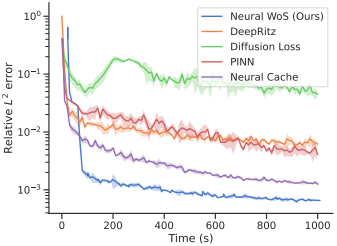
H. C. Nam, J. Berner, and A. Anandkumar. Solving Poisson equations using neural walk-on-spheres. In ICML, 2024

SDE-based neural PDE solver - Results

- We solve (parametric) PDEs from optimal control, chemistry, and finance with > 50 dims. (**out-of-scope for classical methods**).
- Numerical evidence that our method can **overcome the curse of dimensionality**.
- More **efficient and accurate** than competing methods (e.g., PINNs, Deep Ritz).
- Extensions: Combining SDE-based losses with **neural operators** for the **Navier-Stokes eq.**

	PINO	MCNO
Error ($\nu = 10^{-3}$)	2.42± 0.27	2.03± 0.32
Error ($\nu = 10^{-4}$)	5.28± 0.21	4.31± 0.21
Error ($\nu = 10^{-5}$)	9.34± 0.29	6.16± 0.12
Time (Infer, ms)	4.53	4.53
Time (Train, H)	6.17	1.08
Parameters	1481009	1481009





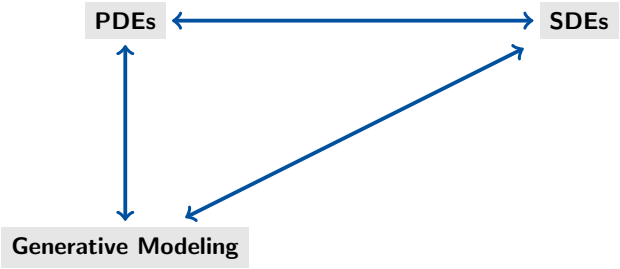
R. Zhang, Q. Meng, R. Zhu, Y. Wang, W. Shi, S. Zhang, Z.-M. Ma, and T.-Y. Liu. Monte Carlo neural operator for learning PDEs via probabilistic representation. arXiv:2302.05104, 2023

J. **Berner**, M. Dablander, and P. Grohs. Numerically solving parametric families of high-dimensional Kolmogorov PDEs via deep learning. In NeurIPS, pages 16615–16627, 2020

L. Richter and J. **Berner**. Robust SDE-based variational formulations for solving linear PDEs via deep learning. In ICML, pages 18649–18666, 2022

H. C. Nam, J. Berner, and A. Anandkumar. Solving Poisson equations using neural walk-on-spheres. In ICML, 2024

SDE-based generative modeling and sampling



Motivation: Selected applications of diffusion models

Image & Video Generation

Fig. 1: https://en.wikipedia.org/wiki/File:Astronaut_Riding_a_Horse_(SDXL).jpg

Motivation: Selected applications of diffusion models

Image & Video Generation

Biology and Chemistry

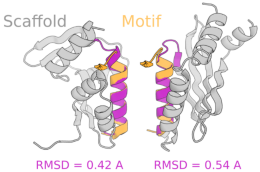


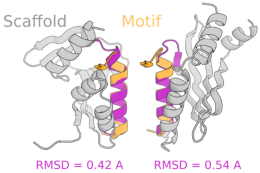
Fig. 2: B. L. Trippe, J. Yim, D. Tischer, T. Broderick, D. Baker, R. Barzilay, and T. Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. arXiv preprint arXiv:2206.04119, 2022

Fig. 1: https://en.wikipedia.org/wiki/File:Astronaut_Riding_a_Horse_(SDXL).jpg

Motivation: Selected applications of diffusion models

Image & Video Generation

Biology and Chemistry



Language modeling

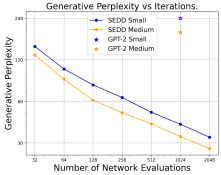


Fig. 3: A. Lou, C. Meng, and S. Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. arXiv preprint arXiv:2310.16834, 2023

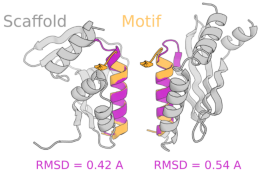
Fig. 2: B. L. Trippe, J. Yim, D. Tischer, T. Broderick, D. Baker, R. Barzilay, and T. Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. arXiv preprint arXiv:2206.04119, 2022

Fig. 1: https://en.wikipedia.org/wiki/File:Astronaut_Riding_a_Horse_(SDXL).jpg

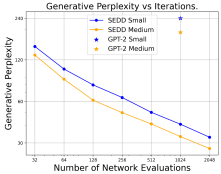
Motivation: Selected applications of diffusion models

Image & Video Generation

Biology and Chemistry



Language modeling



Inverse Problems

Fig. 4: H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye. Diffusion posterior sampling for general noisy inverse problems. arXiv preprint arXiv:2209.14687, 2022

Fig. 3: A. Lou, C. Meng, and S. Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. arXiv preprint arXiv:2310.16834, 2023

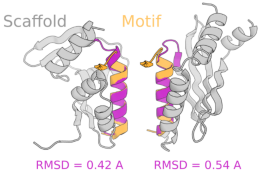
Fig. 2: B. L. Trippe, J. Yim, D. Tischer, T. Broderick, D. Baker, R. Barzilay, and T. Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. arXiv preprint arXiv:2206.04119, 2022

Fig. 1: `https://en.wikipedia.org/wiki/File:Astronaut_Riding_a_Horse_(SDXL).jpg`

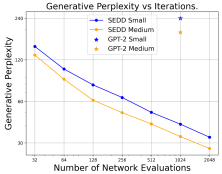
Motivation: Selected applications of diffusion models

Image & Video Generation

Biology and Chemistry



Language modeling



Inverse Problems

Neural Compression

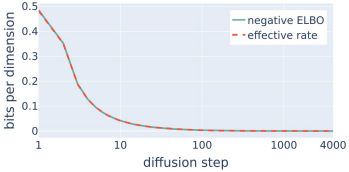


Fig. 5: https://github.com/juliusberner/NeuralCompression

Fig. 4: H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye. Diffusion posterior sampling for general noisy inverse problems. arXiv preprint arXiv:2209.14687, 2022

Fig. 3: A. Lou, C. Meng, and S. Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. arXiv preprint arXiv:2310.16834, 2023

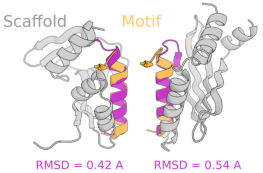
Fig. 2: B. L. Trippe, J. Yim, D. Tischer, T. Broderick, D. Baker, R. Barzilay, and T. Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. arXiv preprint arXiv:2206.04119, 2022

Fig. 1: https://en.wikipedia.org/wiki/File:Astronaut_Riding_a_Horse_(SDXL).jpg

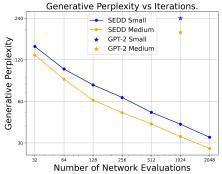
Motivation: Selected applications of diffusion models

Image & Video Generation

Biology and Chemistry

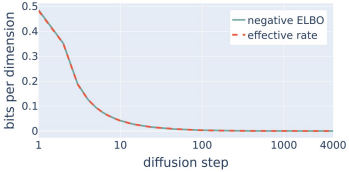


Language modeling



Inverse Problems

Neural Compression



Sampling Problems

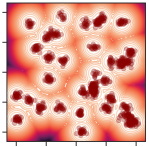


Fig. 6: L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Fig. 5: https://github.com/juliusberner/NeuralCompression

Fig. 4: H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye. Diffusion posterior sampling for general noisy inverse problems. arXiv preprint arXiv:2209.14687, 2022

Fig. 3: A. Lou, C. Meng, and S. Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. arXiv preprint arXiv:2310.16834, 2023

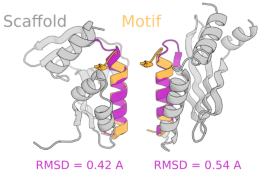
Fig. 2: B. L. Trippe, J. Yim, D. Tischer, T. Broderick, D. Baker, R. Barzilay, and T. Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. arXiv preprint arXiv:2206.04119, 2022

Fig. 1: https://en.wikipedia.org/wiki/File:Astronaut_Riding_a_Horse_(SDXL).jpg

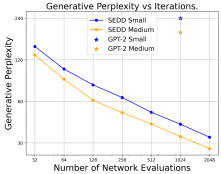
Motivation: Selected applications of diffusion models

Image & Video Generation

Biology and Chemistry

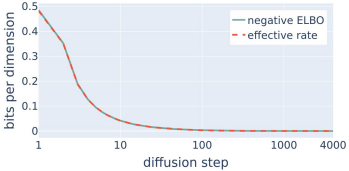


Language modeling



Inverse Problems

Neural Compression



Sampling Problems

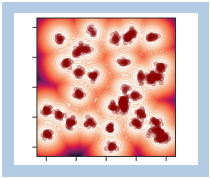


Fig. 6: L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Fig. 5: https://github.com/juliusberner/NeuralCompression

Fig. 4: H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye. Diffusion posterior sampling for general noisy inverse problems. arXiv preprint arXiv:2209.14687, 2022

Fig. 3: A. Lou, C. Meng, and S. Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. arXiv preprint arXiv:2310.16834, 2023

Fig. 2: B. L. Trippe, J. Yim, D. Tischer, T. Broderick, D. Baker, R. Barzilay, and T. Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. arXiv preprint arXiv:2206.04119, 2022

Fig. 1: https://en.wikipedia.org/wiki/File:Astronaut_Riding_a_Horse_(SDXL).jpg

Sampling from high-dimensional distributions

Task: Sample from a high-dimensional distribution p_{target}.

Sampling from high-dimensional distributions

Task: Sample from a high-dimensional distribution p_{target}.

p_{target} can be given in the form of:

1. **samples** $x^{(i)} \sim p_{\text{target}}$ (images, text, sound, ...).

Fig. 1: https://en.m.wikipedia.org/wiki/File:Cat_poster_1.jpg

Sampling from high-dimensional distributions

Task: Sample from a high-dimensional distribution p_{target}.

p_{target} can be given in the form of:

1. **samples** $x^{(i)} \sim p_{\text{target}}$ (images, text, sound, ...).

2. an (unnormalized) **density** q_{target} with $p_{\text{target}} = q_{\text{target}}/\mathcal{Z}$ (e.g., in Bayesian statistics, comp. physics, and chemistry).

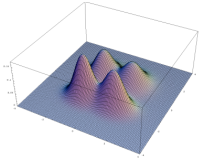


Fig. 2: https://en.wikipedia.org/wiki/File:Bimodal-bivariate-small.png
Fig. 1: https://en.m.wikipedia.org/wiki/File:Cat_poster_1.jpg

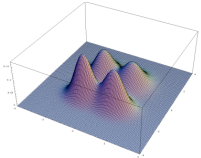
Sampling from high-dimensional distributions

Task: Sample from a high-dimensional distribution p_{target}.

p_{target} can be given in the form of:

1. **samples** $x^{(i)} \sim p_{\text{target}}$ (images, text, sound, ...).

2. an (unnormalized) **density** q_{target} with $p_{\text{target}} = q_{\text{target}}/\mathcal{Z}$ (e.g., in Bayesian statistics, comp. physics, and chemistry).



Generative modeling

Fig. 2: https://en.wikipedia.org/wiki/File:Bimodal-bivariate-small.png
Fig. 1: https://en.m.wikipedia.org/wiki/File:Cat_poster_1.jpg

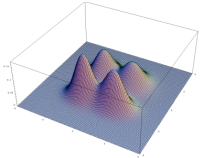
Sampling from high-dimensional distributions

Task: Sample from a high-dimensional distribution p_{target}.

p_{target} can be given in the form of:

1. **samples** $x^{(i)} \sim p_{\text{target}}$ (images, text, sound, ...).

2. an (unnormalized) **density** q_{target} with $p_{\text{target}} = q_{\text{target}}/\mathcal{Z}$ (e.g., in Bayesian statistics, comp. physics, and chemistry).



Generative modeling

Sampling problems

Fig. 2: https://en.wikipedia.org/wiki/File:Bimodal-bivariate-small.png
Fig. 1: https://en.m.wikipedia.org/wiki/File:Cat_poster_1.jpg

Overview of generative models

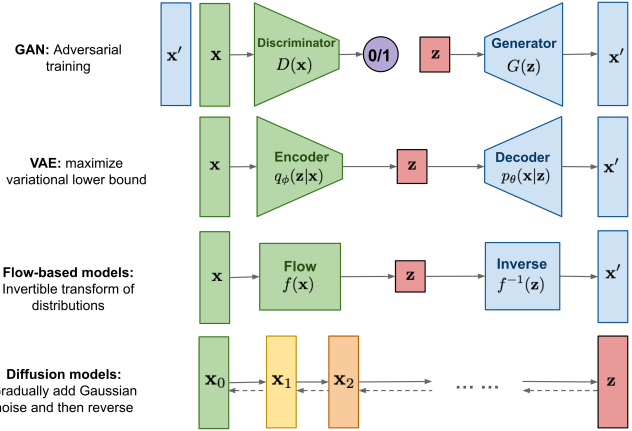
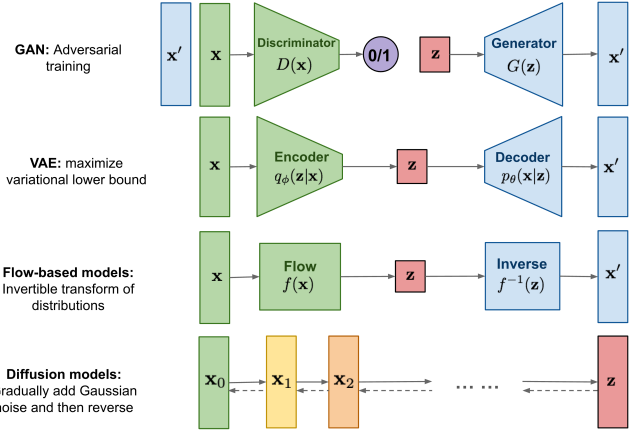


Fig.: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

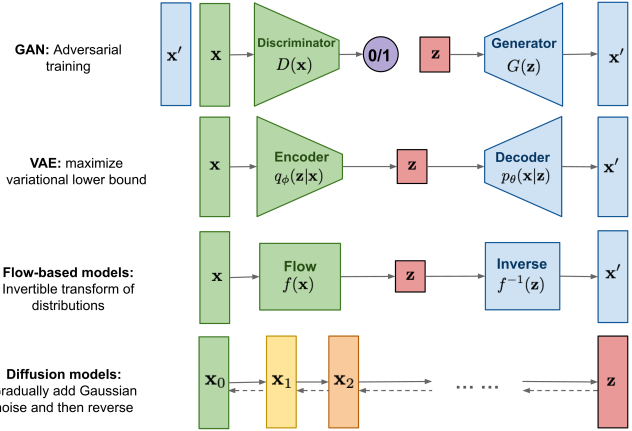
Overview of generative models



Connections: Diffusion models give rise to continuous-time normalizing flows and can be viewed as hierarchical VAEs with fixed encoder.

Fig.: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Overview of generative models



Connections: Diffusion models give rise to continuous-time normalizing flows and can be viewed as hierarchical VAEs with fixed encoder.

Tasks: Flow-based models have been used for Tasks 1 & 2. We show that this is also possible for diffusion models.

Fig.: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Problems

- We cannot use **denoising score matching** (or other adaptations of likelihood training / forward-KL divergence):

$$\mathbb{E}_{x \sim p_{\text{target}},\, n \sim \mathcal{N}(0, \sigma^2 \mathrm{I})} \left[\left\| u_\theta(x + n, \sigma) - x \right\|^2 \right]$$

Problems

- We cannot use **denoising score matching** (or other adaptations of likelihood training / forward-KL divergence):

$$\mathbb{E}_{x \sim p_{\text{target}},\, n \sim \mathcal{N}(0, \sigma^2 \mathrm{I})} \left[\left\| u_\theta(x + n, \sigma) - x \right\|^2 \right]$$

💡 Requires samples $x \sim p_{\text{target}}$ from the target distribution.

Problems

- We cannot use **denoising score matching** (or other adaptations of likelihood training / forward-KL divergence):

$$\mathbb{E}_{x \sim p_{\text{target}},\, n \sim \mathcal{N}(0, \sigma^2 \mathrm{I})} \left[\left\| u_\theta(x + n, \sigma) - x \right\|^2 \right]$$

  - Requires samples $x \sim p_{\text{target}}$ from the target distribution.
  - Importance sampling does not scale to high dims.

Problems

- We cannot use **denoising score matching** (or other adaptations of likelihood training / forward-KL divergence):

$$\mathbb{E}_{x \sim p_{\text{target}},\, n \sim \mathcal{N}(0, \sigma^2 \mathrm{I})} \left[\left\| u_\theta(x + n, \sigma) - x \right\|^2 \right]$$

  - Requires samples $x \sim p_{\text{target}}$ from the target distribution.
  - Importance sampling does not scale to high dims.

- We cannot use **reverse-KL divergence** of normalizing flows:

$$\min_{\mu} D_{\mathrm{KL}}(p_{X_T} | p_{\text{target}})$$

Problems

- We cannot use **denoising score matching** (or other adaptations of likelihood training / forward-KL divergence):

$$\mathbb{E}_{x \sim p_{\text{target}},\, n \sim \mathcal{N}(0, \sigma^2 \mathrm{I})} \left[\left\| u_\theta(x + n, \sigma) - x \right\|^2 \right]$$

  - Requires samples $x \sim p_{\text{target}}$ from the target distribution.
  - Importance sampling does not scale to high dims.

- We cannot use **reverse-KL divergence** of normalizing flows:

$$\min_{\mu} D_{\mathrm{KL}}(p_{X_T} | p_{\text{target}})$$

  - The density p_{X_T} can only be evaluated for ODEs, i.e., $\mathrm{d}X_s = \mu(X_s)\,\mathrm{d}s$.

Problems

- We cannot use **denoising score matching** (or other adaptations of likelihood training / forward-KL divergence):

$$\mathbb{E}_{x\sim p_{\text{target}},\, n\sim\mathcal{N}(0,\sigma^2 \mathrm{I})}\left[\left\|u_\theta(x+n,\sigma)-x\right\|^2\right]$$

  - Requires samples $x \sim p_{\text{target}}$ from the target distribution.
  - Importance sampling does not scale to high dims.

- We cannot use **reverse-KL divergence** of normalizing flows:

$$\min_{\mu} D_{\mathrm{KL}}(p_{X_T} | p_{\text{target}})$$

  - The density p_{X_T} can only be evaluated for ODEs, i.e., $\mathrm{d}X_s = \mu(X_s)\,\mathrm{d}s$.
  - Cannot use the probability flow ODE (score $\nabla \log p_X$ not available during training).

Problems

- We cannot use **denoising score matching** (or other adaptations of likelihood training / forward-KL divergence):

$$\mathbb{E}_{x\sim p_{\text{target}},\, n\sim\mathcal{N}(0,\sigma^2 \mathrm{I})}\left[\left\|u_\theta(x+n,\sigma)-x\right\|^2\right]$$

  - Requires samples $x \sim p_{\text{target}}$ from the target distribution.
  - Importance sampling does not scale to high dims.

- We cannot use **reverse-KL divergence** of normalizing flows:

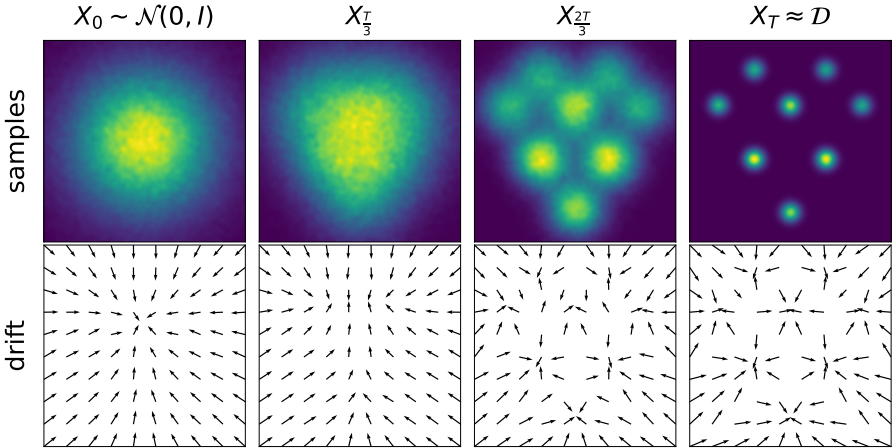
$$\boxed{\min_\mu D_{\mathrm{KL}}(p_{X_T}|p_{\text{target}})}$$

  - The density p_{X_T} can only be evaluated for ODEs, i.e., $\mathrm{d}X_s = \mu(X_s)\,\mathrm{d}s$.
  - Cannot use the probability flow ODE (score $\nabla \log p_X$ not available during training).

Spoiler: Take the divergence on the **path space** $C([0,T],\mathbb{R}^d)$ rather than the target space $\mathbb{R}^d$, i.e., $D_{\mathrm{KL}}(\mathbb{P}_{X^u}|\mathbb{P}_{\text{target}})$.

SDE-based generative modeling and sampling

Goal: Learn the drift μ of a SDE X such that X_T approximates a given distribution.



Connections to PDEs

Goal: $X_T \sim p_{\text{target}}$ with **generative SDE**

$$\mathrm{d}X_s = \mu(X_s, s)\,\mathrm{d}s + \sigma\,\mathrm{d}B_s, \quad X_0 \sim p_{\text{prior}}.$$

$\sigma = 0$: continuous-time **normalizing flow**.

Connections to PDEs

Goal: $X_T \sim p_{\text{target}}$ with **generative SDE**

$$\mathrm{d}X_s = \mu(X_s, s)\,\mathrm{d}s + \sigma\,\mathrm{d}B_s, \quad X_0 \sim p_{\text{prior}}.$$

$\sigma = 0$: continuous-time **normalizing flow**.

Fokker-Planck equation for density p_X:

$$\partial_t p_X = -\operatorname{div}(p_X \mu) + \tfrac{1}{2}\sigma^2 \Delta p_X.$$

$\sigma = 0$: **continuity equation**.

Connections to PDEs

Goal: $X_T \sim p_{\text{target}}$ with **generative SDE**

$$\mathrm{d}X_s = \mu(X_s, s)\,\mathrm{d}s + \sigma\,\mathrm{d}B_s, \quad X_0 \sim p_{\text{prior}}.$$

$\sigma = 0$: continuous-time **normalizing flow**.

Fokker-Planck equation for density p_X:

$$\partial_t p_X = -\operatorname{div}(p_X \mu) + \tfrac{1}{2}\sigma^2 \Delta p_X.$$

$\sigma = 0$: **continuity equation**.

Regularization: minimize energy

$$\inf_{\mu} \int_0^T \mathbb{E}\left[\|\mu(X_s, s)\|^2\right] \mathrm{d}s.$$

$\sigma > 0$: Dynamic **Schrödinger bridge**.

$\sigma = 0$: **Optimal** W_2**-transport**.

J.-D. Benamou and Y. Brenier. A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. Numerische Mathematik, 84(3):375–393, 2000

Connections to PDEs

Goal: $X_T \sim p_{\text{target}}$ with **generative SDE**

$$\mathrm{d}X_s = \mu(X_s, s)\,\mathrm{d}s + \sigma\,\mathrm{d}B_s, \quad X_0 \sim p_{\text{prior}}.$$

$\sigma = 0$: continuous-time **normalizing flow**.

Fokker-Planck equation for density p_X:

$$\partial_t p_X = -\operatorname{div}(p_X \mu) + \tfrac{1}{2}\sigma^2 \Delta p_X.$$

$\sigma = 0$: **continuity equation**.

Regularization: minimize energy

$$\inf_{\mu} \int_0^T \mathbb{E}\left[\|\mu(X_s, s)\|^2\right] \mathrm{d}s.$$

$\sigma > 0$: Dynamic **Schrödinger bridge**.

$\sigma = 0$: **Optimal W_2-transport**.

HJB equation for $\mu = -\nabla\Phi$:

$$\partial_t \Phi = \tfrac{1}{2}\|\nabla\Phi\|^2 - \tfrac{1}{2}\sigma^2 \Delta\Phi.$$

J.-D. Benamou and Y. Brenier. A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. Numerische Mathematik, 84(3):375–393, 2000

Connections to PDEs

Goal: $X_T \sim p_{\text{target}}$ with **generative SDE**

$$\mathrm{d}X_s = \mu(X_s, s)\,\mathrm{d}s + \sigma\,\mathrm{d}B_s, \quad X_0 \sim p_{\text{prior}}.$$

$\sigma = 0$: continuous-time **normalizing flow**.

Fokker-Planck equation for density p_X:

$$\partial_t p_X = -\operatorname{div}(p_X \mu) + \tfrac{1}{2}\sigma^2 \Delta p_X.$$

$\sigma = 0$: **continuity equation**.

Regularization: minimize energy

$$\inf_{\mu} \int_0^T \mathbb{E}\left[\|\mu(X_s, s)\|^2\right] \mathrm{d}s.$$

$\sigma > 0$: Dynamic **Schrödinger bridge**.
$\sigma = 0$: **Optimal** W_2**-transport**.

HJB equation for $\mu = -\nabla\Phi$:

$$\partial_t \Phi = \tfrac{1}{2}\|\nabla\Phi\|^2 - \tfrac{1}{2}\sigma^2 \Delta\Phi.$$

💡 Losses of neural PDE solvers have been used as regularizers.

T. Koshizuka and I. Sato. Neural Lagrangian Schrödinger bridge. In ICLR, 2022

L. Ruthotto, S. J. Osher, et al. A machine learning framework for solving high-dimensional mean field game and mean field control problems. Proceedings of the National Academy of Sciences, 117(17):9183–9193, 2020

L. Yang and G. E. Karniadakis. Potential flow generator with L^2 optimal transport regularity for generative models. IEEE Transactions on Neural Networks and Learning Systems, 33(2):528–538, 2020

D. Onken, S. W. Fung, X. Li, and L. Ruthotto. OT-Flow: Fast and accurate continuous normalizing flows via optimal transport. In Proceedings of the AAAI Conference on Artificial Intelligence, number 10, pages 9223–9232, 2021

J.-D. Benamou and Y. Brenier. A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. Numerische Mathematik, 84(3):375–393, 2000

Connections to PDEs

Goal: $X_T \sim p_{\text{target}}$ with **generative SDE**

$$\mathrm{d}X_s = \mu(X_s, s)\,\mathrm{d}s + \sigma\,\mathrm{d}B_s, \quad X_0 \sim p_{\text{prior}}.$$

$\sigma = 0$: continuous-time **normalizing flow**.

Fokker-Planck equation for density p_X:

$$\partial_t p_X = -\operatorname{div}(p_X \mu) + \tfrac{1}{2}\sigma^2 \Delta p_X.$$

$\sigma = 0$: **continuity equation**.

Regularization: minimize energy

$$\inf_{\mu} \int_0^T \mathbb{E}\left[\|\mu(X_s, s)\|^2\right] \mathrm{d}s.$$

$\sigma > 0$: Dynamic **Schrödinger bridge**.
$\sigma = 0$: **Optimal W_2-transport**.

HJB equation for $\mu = -\nabla\Phi$:

$$\partial_t \Phi = \tfrac{1}{2}\|\nabla\Phi\|^2 - \tfrac{1}{2}\sigma^2 \Delta\Phi.$$

💡 Losses of neural PDE solvers have been used as regularizers.
💡 Use neural PDE solver for the (log-transformed) Fokker-Planck eq. with $p_X(\cdot, T) = q_{\text{target}}$ to tackle **sampling problems**

J. Sun, J. Berner, K. Azizzadenesheli, and A. Anandkumar. Physics-informed neural networks for sampling. In ICLR 2024 Workshop on AI4DifferentialEquations In Science, 2024

T. Koshizuka and I. Sato. Neural Lagrangian Schrödinger bridge. In ICLR, 2022

L. Ruthotto, S. J. Osher, et al. A machine learning framework for solving high-dimensional mean field game and mean field control problems. Proceedings of the National Academy of Sciences, 117(17):9183–9193, 2020

L. Yang and G. E. Karniadakis. Potential flow generator with L^2 optimal transport regularity for generative models. IEEE Transactions on Neural Networks and Learning Systems, 33(2):528–538, 2020

D. Onken, S. W. Fung, X. Li, and L. Ruthotto. OT-Flow: Fast and accurate continuous normalizing flows via optimal transport. In Proceedings of the AAAI Conference on Artificial Intelligence, number 10, pages 9223–9232, 2021

J.-D. Benamou and Y. Brenier. A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. Numerische Mathematik, 84(3):375–393, 2000

Connections to PDEs

Goal: $X_T \sim p_{\text{target}}$ with **generative SDE**

$$\mathrm{d}X_s = \mu(X_s, s)\,\mathrm{d}s + \sigma\,\mathrm{d}B_s, \quad X_0 \sim p_{\text{prior}}.$$

$\sigma = 0$: continuous-time **normalizing flow**.

Fokker-Planck equation for density p_X:

$$\partial_t p_X = -\operatorname{div}(p_X \mu) + \tfrac{1}{2}\sigma^2 \Delta p_X.$$

$\sigma = 0$: **continuity equation**.

Regularization: minimize energy

$$\inf_{\mu} \int_0^T \mathbb{E}\left[\|\mu(X_s, s)\|^2\right] \mathrm{d}s.$$

$\sigma > 0$: Dynamic **Schrödinger bridge**.
$\sigma = 0$: **Optimal** W_2**-transport**.

HJB equation for $\mu = -\nabla\Phi$:

$$\partial_t \Phi = \tfrac{1}{2}\|\nabla\Phi\|^2 - \tfrac{1}{2}\sigma^2 \Delta\Phi.$$

💡 Losses of neural PDE solvers have been used as regularizers.
💡 Use neural PDE solver for the (log-transformed) Fokker-Planck eq. with $p_X(\cdot, T) = q_{\text{target}}$ to tackle **sampling problems** — but no clear benefit of diffusion-based models (i.e., $\sigma > 0$).

J. Sun, J. Berner, K. Azizzadenesheli, and A. Anandkumar. Physics-informed neural networks for sampling. In ICLR 2024 Workshop on AI4DifferentialEquations In Science, 2024

T. Koshizuka and I. Sato. Neural Lagrangian Schrödinger bridge. In ICLR, 2022

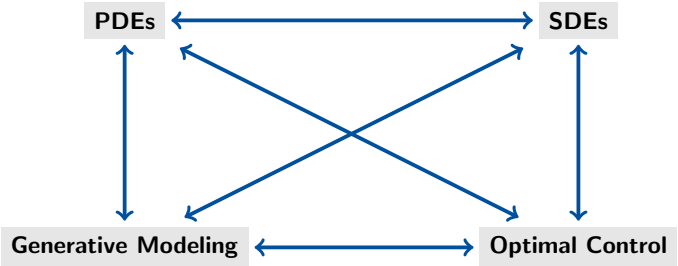
L. Ruthotto, S. J. Osher, et al. A machine learning framework for solving high-dimensional mean field game and mean field control problems. Proceedings of the National Academy of Sciences, 117(17):9183–9193, 2020

L. Yang and G. E. Karniadakis. Potential flow generator with L^2 optimal transport regularity for generative models. IEEE Transactions on Neural Networks and Learning Systems, 33(2):528–538, 2020

D. Onken, S. W. Fung, X. Li, and L. Ruthotto. OT-Flow: Fast and accurate continuous normalizing flows via optimal transport. In Proceedings of the AAAI Conference on Artificial Intelligence, number 10, pages 9223–9232, 2021

J.-D. Benamou and Y. Brenier. A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. Numerische Mathematik, 84(3):375–393, 2000

Diffusion models, Schrödinger bridges, and optimal control

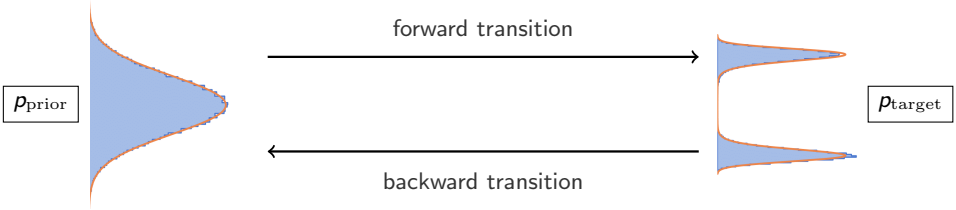


Framework: Time-reversals

General setting: Forward & backward transitions between prior p_{prior} and target p_{target}.

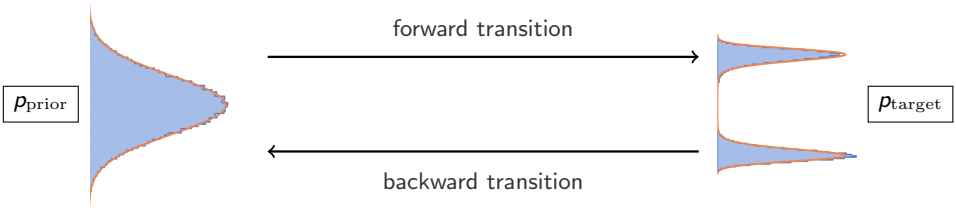
Framework: Time-reversals

General setting: Forward & backward transitions between prior p_{prior} and target p_{target}.



Framework: Time-reversals

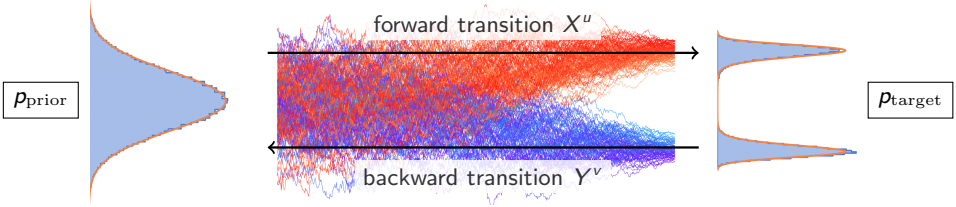
General setting: Forward & backward transitions between prior p_{prior} and target p_{target}.



Examples: VI in latent variable models (VAEs), (optimal) transport, (Schrödinger) bridges.

Framework: Time-reversals

General setting: Forward & backward transitions between prior p_{prior} and target p_{target}.



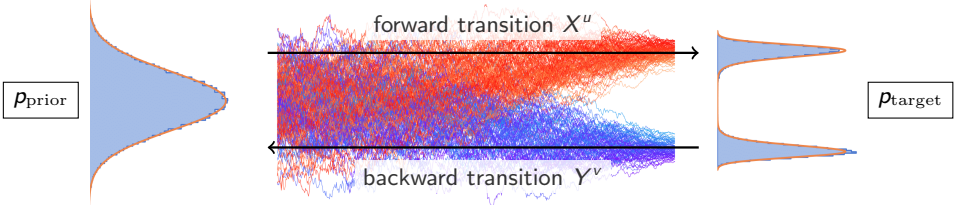
Examples: VI in latent variable models (VAEs), (optimal) transport, (Schrödinger) bridges.

Our setting: Controlled SDEs (notation: $\overleftarrow{\sigma}(t) := \sigma(T-t)$)

$$\begin{aligned} \mathrm{d}X_s^u &= \sigma(s)u(X_s^u, s)\,\mathrm{d}s + \sigma(s)\,\mathrm{d}W_s, \quad X_0 \sim p_{\text{prior}}, \\ \mathrm{d}Y_s^v &= \overleftarrow{\sigma}(s)\overleftarrow{v}(Y_s^v, s)\,\mathrm{d}s + \overleftarrow{\sigma}(s)\,\mathrm{d}W_s, \quad Y_0 \sim p_{\text{target}}. \end{aligned}$$

Framework: Time-reversals

General setting: Forward & backward transitions between prior p_{prior} and target p_{target}.



Examples: VI in latent variable models (VAEs), (optimal) transport, (Schrödinger) bridges.

Our setting: Controlled SDEs (notation: $\overleftarrow{\sigma}(t) := \sigma(T-t)$)

$$\begin{aligned} \mathrm{d}X_s^u &= \sigma(s)u(X_s^u, s)\,\mathrm{d}s + \sigma(s)\,\mathrm{d}W_s, \quad X_0 \sim p_{\text{prior}}, \\ \mathrm{d}Y_s^v &= \overleftarrow{\sigma}(s)\overleftarrow{v}(Y_s^v, s)\,\mathrm{d}s + \overleftarrow{\sigma}(s)\,\mathrm{d}W_s, \quad Y_0 \sim p_{\text{target}}. \end{aligned}$$

New objective: Find u, v, such that X^u is the time-reversal $\overleftarrow{Y}^v$ of Y^v, i.e., consider the path measures $\mathbb{P}_{X^u}$ and $\mathbb{P}_{\overleftarrow{Y}^v}$, a divergence D, and optimize $\min_{u,v} D\big(\mathbb{P}_{X^u} \big| \mathbb{P}_{\overleftarrow{Y}^v}\big)$.

Unifying objective

How to calculate the divergence $D\left(\mathbb{P}_{X^u} \middle| \mathbb{P}_{\tilde{Y}^v}\right)$?

Unifying objective

How to calculate the divergence $D\big(\mathbb{P}_{X^u} \big| \mathbb{P}_{\tilde{Y}^v}\big)$?

💡 We combine the PDE perspective with tools from optimal control to derive the **Log-likelihood for path measures:**

$$\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\tilde{Y}^v}}(X^w) = \int_0^T \left((u+v) \cdot \left(w + \frac{v-u}{2} \right) + \mathrm{div}(\sigma v) \right)(X_s^w, s)\, \mathrm{d}s + \underbrace{\int_0^T (u+v)(X_s^w, s) \cdot \mathrm{d}W_s}_{\text{zero expectation}} + \log \frac{p_{\mathrm{prior}}(X_0^w)}{p_{\mathrm{target}}(X_T^w)}$$

L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Unifying objective

How to calculate the divergence $D\big(\mathbb{P}_{X^u} \big| \mathbb{P}_{\tilde{Y}^v}\big)$?

💡 We combine the PDE perspective with tools from optimal control to derive the **Log-likelihood for path measures:**

$$\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\tilde{Y}^v}}(X^w) = \int_0^T \left((u+v) \cdot \left(w + \frac{v-u}{2} \right) + \mathrm{div}(\sigma v) \right)(X_s^w, s)\, \mathrm{d}s + \underbrace{\int_0^T (u+v)(X_s^w, s) \cdot \mathrm{d}W_s}_{\text{zero expectation}} + \log \frac{p_{\mathrm{prior}}(X_0^w)}{p_{\mathrm{target}}(X_T^w)}$$

Typical choice $D = D_{\mathrm{KL}}$.

L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Unifying objective

How to calculate the divergence $D\big(\mathbb{P}_{X^u} \big| \mathbb{P}_{\tilde{Y}^v}\big)$?

💡 We combine the PDE perspective with tools from optimal control to derive the **Log-likelihood for path measures:**

$$\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\tilde{Y}^v}}(X^w) = \int_0^T \left((u+v)\cdot\left(w + \frac{v-u}{2}\right) + \mathrm{div}(\sigma v)\right)(X_s^w, s)\,\mathrm{d}s + \underbrace{\int_0^T (u+v)(X_s^w, s)\cdot \mathrm{d}W_s}_{\text{zero expectation}} + \log \frac{p_{\mathrm{prior}}(X_0^w)}{p_{\mathrm{target}}(X_T^w)}$$

Typical choice $D = D_{\mathrm{KL}}$. **Forward-KL** recovers objectives from generative modeling:

- "Likelihood Schrödinger bridge" (non-unique)

T. Chen, G.-H. Liu, and E. A. Theodorou. Likelihood training of Schrödinger Bridge using Forward-Backward SDEs theory. arXiv preprint arXiv:2110.11291, 2021

L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Unifying objective

How to calculate the divergence $D\big(\mathbb{P}_{X^u} \big| \mathbb{P}_{\tilde{Y}^v}\big)$?

We combine the PDE perspective with tools from optimal control to derive the **Log-likelihood for path measures:**

$$\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\tilde{Y}^v}}(X^w) = \int_0^T \left((u+v)\cdot\left(w + \frac{v-u}{2}\right) + \mathrm{div}(\sigma v)\right)(X_s^w, s)\,\mathrm{d}s + \underbrace{\int_0^T (u+v)(X_s^w, s)\cdot \mathrm{d}W_s}_{\text{zero expectation}} + \log \frac{p_{\mathrm{prior}}(X_0^w)}{p_{\mathrm{target}}(X_T^w)}$$

Typical choice $D = D_{\mathrm{KL}}$. **Forward-KL** recovers objectives from generative modeling:

- "Likelihood Schrödinger bridge" (non-unique)
- Fixed v: ELBO for diffusion models

T. Chen, G.-H. Liu, and E. A. Theodorou. Likelihood training of Schrödinger Bridge using Forward-Backward SDEs theory. arXiv preprint arXiv:2110.11291, 2021

L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Unifying objective

Sampling problems: Consider the **reverse-KL** $D_{\mathrm{KL}}\left(\mathbb{P}_{X^u} | \mathbb{P}_{\tilde{Y}^v}\right) = \mathbb{E}\left[\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\tilde{Y}^v}}(X^u)\right].$

Unifying objective

Sampling problems: Consider the **reverse-KL** $D_{\mathrm{KL}}\left(\mathbb{P}_{X^u} | \mathbb{P}_{\tilde{Y}^v}\right) = \mathbb{E}\left[\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\tilde{Y}^v}}(X^u)\right].$

Bridge sampler:

$$D_{\mathrm{KL}}\left(\mathbb{P}_{X^u} | \mathbb{P}_{\tilde{Y}^v}\right) = \mathbb{E}\left[\int_0^T \underbrace{\left(\tfrac{1}{2}\|u+v\|^2 + \mathrm{div}(\sigma v)\right)(X_s^u, s)}_{\text{running cost}} \,\mathrm{d}s + \underbrace{\log \frac{p_{\mathrm{prior}}(X_0^u)}{q_{\mathrm{target}}(X_T^u)}}_{\text{initial / terminal cost}} + \underbrace{\log Z}_{\text{const.}}\right]$$

Unifying objective

Sampling problems: Consider the **reverse-KL** $D_{\mathrm{KL}}\left(\mathbb{P}_{X^u} | \mathbb{P}_{\overleftarrow{Y}^v}\right) = \mathbb{E}\left[\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\overleftarrow{Y}^v}}(X^u)\right].$

Bridge sampler:

$$D_{\mathrm{KL}}\left(\mathbb{P}_{X^u} | \mathbb{P}_{\overleftarrow{Y}^v}\right) = \mathbb{E}\left[\int_0^T \underbrace{\left(\tfrac{1}{2}\|u+v\|^2 + \mathrm{div}(\sigma v)\right)(X_s^u, s)}_{\text{running cost}} \,\mathrm{d}s + \underbrace{\log \frac{p_{\mathrm{prior}}(X_0^u)}{q_{\mathrm{target}}(X_T^u)}}_{\text{initial / terminal cost}} + \underbrace{\log Z}_{\text{const.}}\right]$$

We can also derive unique objectives:

Unifying objective

Sampling problems: Consider the **reverse-KL** $D_{\mathrm{KL}}\left(\mathbb{P}_{X^u} | \mathbb{P}_{\tilde{Y}^v}\right) = \mathbb{E}\left[\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\tilde{Y}^v}}(X^u)\right].$

Bridge sampler:

$$D_{\mathrm{KL}}\left(\mathbb{P}_{X^u} | \mathbb{P}_{\tilde{Y}^v}\right) = \mathbb{E}\left[\int_0^T \underbrace{\left(\tfrac{1}{2}\|u+v\|^2 + \mathrm{div}(\sigma v)\right)(X_s^u, s)}_{\text{running cost}} \,\mathrm{d}s + \underbrace{\log \frac{p_{\mathrm{prior}}(X_0^u)}{q_{\mathrm{target}}(X_T^u)}}_{\text{initial / terminal cost}} + \underbrace{\log Z}_{\text{const.}}\right]$$

We can also derive unique objectives:

- Fixed v: Time-Reversed Diffusion Sampler (DIS).

J. **Berner**, L. Richter, and K. Ullrich. An optimal control perspective on diffusion-based generative modeling. In Transactions on Machine Learning Research, 2024

Unifying objective

Sampling problems: Consider the **reverse-KL** $D_{\mathrm{KL}}(\mathbb{P}_{X^u}|\mathbb{P}_{\tilde{Y}^v}) = \mathbb{E}\left[\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\tilde{Y}^v}}(X^u)\right].$

Bridge sampler:

$$D_{\mathrm{KL}}(\mathbb{P}_{X^u}|\mathbb{P}_{\tilde{Y}^v}) = \mathbb{E}\left[\int_0^T \underbrace{\left(\tfrac{1}{2}\|u+v\|^2 + \mathrm{div}(\sigma v)\right)(X_s^u, s)}_{\text{running cost}}\,\mathrm{d}s + \underbrace{\log \frac{p_{\mathrm{prior}}(X_0^u)}{q_{\mathrm{target}}(X_T^u)}}_{\text{initial / terminal cost}} + \underbrace{\log Z}_{\text{const.}}\right]$$

We can also derive unique objectives:

- Fixed v: Time-Reversed Diffusion Sampler (DIS).
- Reference process: Path Integral Sampler (PIS) & Denoising Diffusion Sampler (DDS).

Q. Zhang and Y. Chen. Path Integral Sampler: a stochastic control approach for sampling. In ICLR, 2022

F. Vargas, W. Grathwohl, and A. Doucet. Denoising diffusion samplers. In ICLR, 2023

J. **Berner**, L. Richter, and K. Ullrich. An optimal control perspective on diffusion-based generative modeling. In Transactions on Machine Learning Research, 2024

Unifying objective

Sampling problems: Consider the **reverse-KL** $D_{\mathrm{KL}}\left(\mathbb{P}_{X^u} | \mathbb{P}_{\tilde{Y}^v}\right) = \mathbb{E}\left[\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\tilde{Y}^v}}(X^u)\right].$

Bridge sampler:

$$D_{\mathrm{KL}}\left(\mathbb{P}_{X^u} | \mathbb{P}_{\tilde{Y}^v}\right) = \mathbb{E}\left[\int_0^T \underbrace{\left(\tfrac{1}{2}\|u+v\|^2 + \mathrm{div}(\sigma v)\right)(X_s^u, s)}_{\text{running cost}} \,\mathrm{d}s + \underbrace{\log \frac{p_{\mathrm{prior}}(X_0^u)}{q_{\mathrm{target}}(X_T^u)}}_{\text{initial / terminal cost}} + \underbrace{\log Z}_{\text{const.}}\right]$$

We can also derive unique objectives:

- Fixed v: Time-Reversed Diffusion Sampler (DIS).
- Reference process: Path Integral Sampler (PIS) & Denoising Diffusion Sampler (DDS).

Short-comings:

Q. Zhang and Y. Chen. Path Integral Sampler: a stochastic control approach for sampling. In ICLR, 2022

F. Vargas, W. Grathwohl, and A. Doucet. Denoising diffusion samplers. In ICLR, 2023

J. **Berner**, L. Richter, and K. Ullrich. An optimal control perspective on diffusion-based generative modeling. In Transactions on Machine Learning Research, 2024

Unifying objective

Sampling problems: Consider the **reverse-KL** $D_{\mathrm{KL}}\left(\mathbb{P}_{X^u} | \mathbb{P}_{\tilde{Y}^v}\right) = \mathbb{E}\left[\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\tilde{Y}^v}}(X^u)\right].$

Bridge sampler:

$$D_{\mathrm{KL}}\left(\mathbb{P}_{X^u} | \mathbb{P}_{\tilde{Y}^v}\right) = \mathbb{E}\left[\int_0^T \underbrace{\left(\tfrac{1}{2}\|u+v\|^2 + \mathrm{div}(\sigma v)\right)(X_s^u, s)}_{\text{running cost}} \,\mathrm{d}s + \underbrace{\log \frac{p_{\mathrm{prior}}(X_0^u)}{q_{\mathrm{target}}(X_T^u)}}_{\text{initial / terminal cost}} + \underbrace{\log Z}_{\text{const.}}\right]$$

We can also derive unique objectives:

- Fixed v: Time-Reversed Diffusion Sampler (DIS).
- Reference process: Path Integral Sampler (PIS) & Denoising Diffusion Sampler (DDS).

Short-comings:

👎 **No exploration:** The simulation is "on-policy" (with the learned model u).

Q. Zhang and Y. Chen. Path Integral Sampler: a stochastic control approach for sampling. In ICLR, 2022

F. Vargas, W. Grathwohl, and A. Doucet. Denoising diffusion samplers. In ICLR, 2023

J. Berner, L. Richter, and K. Ullrich. An optimal control perspective on diffusion-based generative modeling. In Transactions on Machine Learning Research, 2024

Unifying objective

Sampling problems: Consider the **reverse-KL** $D_{\mathrm{KL}}\left(\mathbb{P}_{X^u} | \mathbb{P}_{\tilde{Y}^v}\right) = \mathbb{E}\left[\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\tilde{Y}^v}}(X^u)\right].$

Bridge sampler:

$$D_{\mathrm{KL}}\left(\mathbb{P}_{X^u} | \mathbb{P}_{\tilde{Y}^v}\right) = \mathbb{E}\left[\int_0^T \underbrace{\left(\tfrac{1}{2}\|u+v\|^2 + \mathrm{div}(\sigma v)\right)(X_s^u, s)}_{\text{running cost}} \,\mathrm{d}s + \underbrace{\log \frac{p_{\mathrm{prior}}(X_0^u)}{q_{\mathrm{target}}(X_T^u)}}_{\text{initial / terminal cost}} + \underbrace{\log Z}_{\text{const.}}\right]$$

We can also derive unique objectives:

- Fixed v: Time-Reversed Diffusion Sampler (DIS).
- Reference process: Path Integral Sampler (PIS) & Denoising Diffusion Sampler (DDS).

Short-comings:

👎 **No exploration:** The simulation is "on-policy" (with the learned model u).

👎 **Inefficient:** Requires to backpropagate through the SDE solver.

Q. Zhang and Y. Chen. Path Integral Sampler: a stochastic control approach for sampling. In ICLR, 2022

F. Vargas, W. Grathwohl, and A. Doucet. Denoising diffusion samplers. In ICLR, 2023

J. **Berner**, L. Richter, and K. Ullrich. An optimal control perspective on diffusion-based generative modeling. In Transactions on Machine Learning Research, 2024

Unifying objective

Sampling problems: Consider the **reverse-KL** $D_{\mathrm{KL}}(\mathbb{P}_{X^u}|\mathbb{P}_{\tilde{Y}^v}) = \mathbb{E}\left[\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\tilde{Y}^v}}(X^u)\right].$

Bridge sampler:

$$D_{\mathrm{KL}}(\mathbb{P}_{X^u}|\mathbb{P}_{\tilde{Y}^v}) = \mathbb{E}\left[\int_0^T \underbrace{\left(\tfrac{1}{2}\|u+v\|^2 + \mathrm{div}(\sigma v)\right)(X_s^u, s)}_{\text{running cost}}\,\mathrm{d}s + \underbrace{\log \frac{p_{\mathrm{prior}}(X_0^u)}{q_{\mathrm{target}}(X_T^u)}}_{\text{initial / terminal cost}} + \underbrace{\log Z}_{\text{const.}}\right]$$

We can also derive unique objectives:

- Fixed v: Time-Reversed Diffusion Sampler (DIS).
- Reference process: Path Integral Sampler (PIS) & Denoising Diffusion Sampler (DDS).

Short-comings:

- **No exploration:** The simulation is "on-policy" (with the learned model u).
- **Inefficient:** Requires to backpropagate through the SDE solver.
- **Mode collapse:** Reverse-KL divergence suffers from mode collapse.

Q. Zhang and Y. Chen. Path Integral Sampler: a stochastic control approach for sampling. In ICLR, 2022

F. Vargas, W. Grathwohl, and A. Doucet. Denoising diffusion samplers. In ICLR, 2023

J. **Berner**, L. Richter, and K. Ullrich. An optimal control perspective on diffusion-based generative modeling. In Transactions on Machine Learning Research, 2024

Log-variance divergence

Our framework allows for other divergences: We propose the **log-variance divergence**

$$D_{\mathrm{LV}}^{w}(\mathbb{P}_{X^u}, \mathbb{P}_{\overleftarrow{Y}^v}) = \mathbb{V}\left(\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\overleftarrow{Y}^v}}(X^w)\right).$$

L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Log-variance divergence

💡 Our framework allows for other divergences: We propose the **log-variance divergence**

$$D_{\mathrm{LV}}^{w}(\mathbb{P}_{X^u}, \mathbb{P}_{\overleftarrow{Y}^v}) = \mathbb{V}\left(\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\overleftarrow{Y}^v}}(X^w)\right).$$

👍 **off-policy:** choose w to balance exploration and exploitation.

L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Log-variance divergence

💡 Our framework allows for other divergences: We propose the **log-variance divergence**

$$D_{\mathrm{LV}}^{w}(\mathbb{P}_{X^u}, \mathbb{P}_{\overleftarrow{Y}^v}) = \mathbb{V}\left(\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\overleftarrow{Y}^v}}(X^w)\right).$$

👍 **off-policy:** choose w to balance exploration and exploitation.

👍 **more efficient:** no differentiation through SDE solver.

L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Log-variance divergence

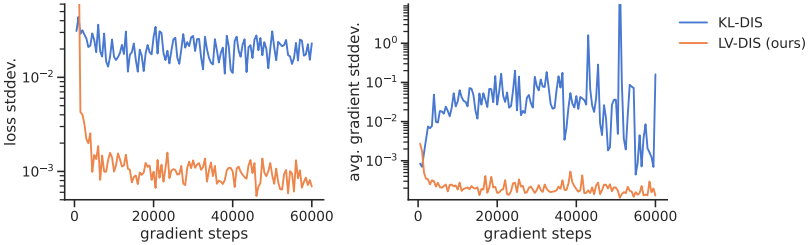
💡 Our framework allows for other divergences: We propose the **log-variance divergence**

$$D_{\mathrm{LV}}^{w}(\mathbb{P}_{X^u}, \mathbb{P}_{\overleftarrow{Y}^v}) = \mathbb{V}\left(\log \frac{\mathrm{d}\mathbb{P}_{X^u}}{\mathrm{d}\mathbb{P}_{\overleftarrow{Y}^v}}(X^w)\right).$$

👍 **off-policy:** choose w to balance exploration and exploitation.

👍 **more efficient:** no differentiation through SDE solver.

👍 **sticking-the-landing:** zero variance at the optimum.



L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Gaussian mixture

Better mode coverage: Improved performance with D_{LV} (compared against D_{KL}) for PIS, DIS, DDS, and the general bridge sampler.

L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Gaussian mixture

Better mode coverage: Improved performance with D_{LV} (compared against D_{KL}) for PIS, DIS, DDS, and the general bridge sampler.

Metrics: Log-norm. const., Sinkhorn distance to ground truth, normalized effective sample size, and average stddev.

Method	Divergence	$\Delta \log \mathcal{Z}(rw) \downarrow$	$\mathcal{W}_\gamma^2 \downarrow$	ESS $\uparrow$	Δstd $\downarrow$
CRAFT		0.012	<u>0.020</u>	-	0.019
PIS	KL	0.249	0.467	0.0051	1.937
	LV	**<u>0.001</u>**	**<u>0.020</u>**	**0.9093**	**0.023**
DIS	KL	**0.015**	0.064	0.0226	2.522
	LV	0.017	**<u>0.020</u>**	**0.8660**	**<u>0.004</u>**
DDS	KL	0.005	0.042	0.0737	2.161
	LV	**<u>0.001</u>**	**<u>0.020</u>**	**0.8929**	**0.006**
Bridge	KL	0.560	0.393	0.0180	0.698
	LV	**0.100**	**<u>0.020</u>**	**<u>0.9669</u>**	**0.010**

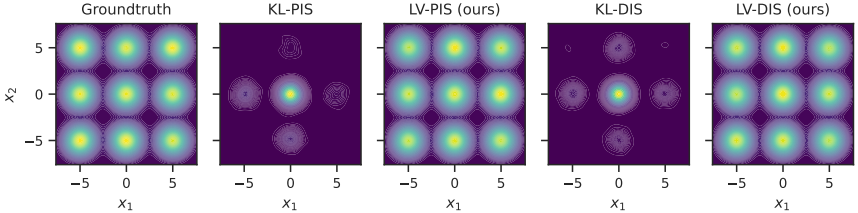
L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Gaussian mixture

Better mode coverage: Improved performance with D_{LV} (compared against D_{KL}) for PIS, DIS, DDS, and the general bridge sampler.

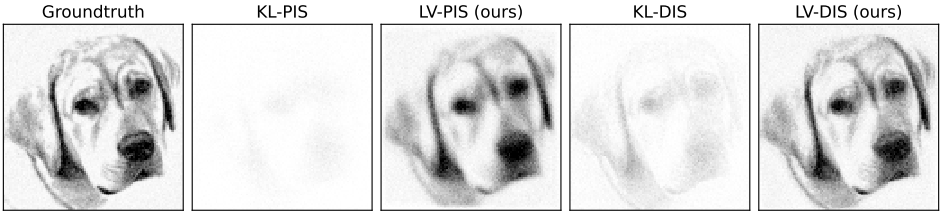
Metrics: Log-norm. const., Sinkhorn distance to ground truth, normalized effective sample size, and average stddev.

Method	Divergence	$\Delta \log \mathcal{Z}(rw) \downarrow$	$\mathcal{W}_\gamma^2 \downarrow$	ESS $\uparrow$	Δstd $\downarrow$
CRAFT		0.012	<u>0.020</u>	-	0.019
PIS	KL	0.249	0.467	0.0051	1.937
	LV	**<u>0.001</u>**	**<u>0.020</u>**	**0.9093**	**0.023**
DIS	KL	**0.015**	0.064	0.0226	2.522
	LV	0.017	**<u>0.020</u>**	**0.8660**	**<u>0.004</u>**
DDS	KL	0.005	0.042	0.0737	2.161
	LV	**<u>0.001</u>**	**<u>0.020</u>**	**0.8929**	**0.006**
Bridge	KL	0.560	0.393	0.0180	0.698
	LV	**0.100**	**<u>0.020</u>**	**<u>0.9669</u>**	**0.010**



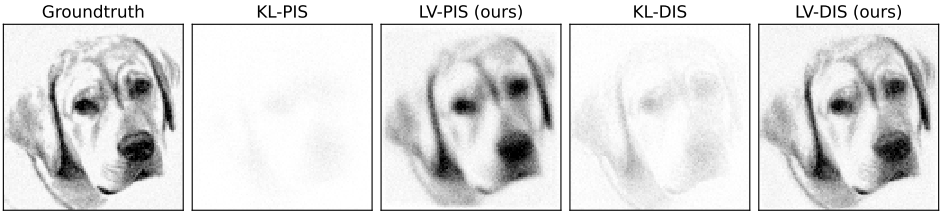
L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Image density and funnel distribution



L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Image density and funnel distribution

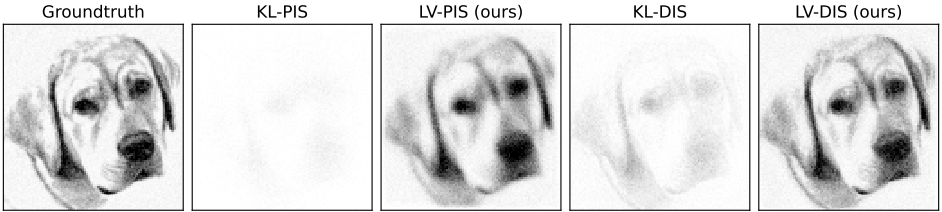


Funnel distribution (challenging benchmark for sampling methods):

$$p_{\text{target}}(x) = \mathcal{N}(x_1; 0, 9) \prod_{i=2}^{10} \mathcal{N}(x_i; 0, e^{x_1})$$

L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Image density and funnel distribution

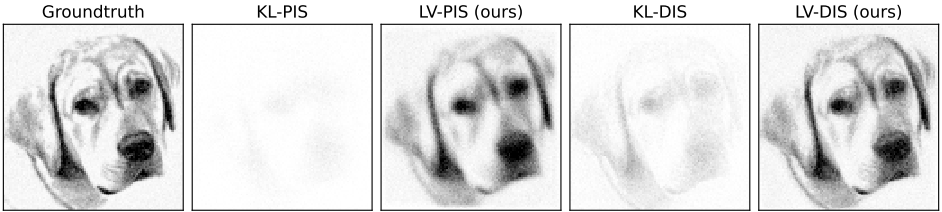


Funnel distribution (challenging benchmark for sampling methods):

$$p_{\text{target}}(x) = \mathcal{N}(x_1; 0, 9) \prod_{i=2}^{10} \mathcal{N}(x_i; 0, e^{x_1})$$

Method	Divergence	$\Delta \log \mathcal{Z}(rw) \downarrow$	$\mathcal{W}_\gamma^2 \downarrow$	ESS $\uparrow$	Δstd $\downarrow$
CRAFT		0.123	5.517	-	6.139
PIS	KL	0.111	5.639	**0.1333**	6.921
	LV	**0.097**	**5.593**	0.0746	**6.852**
DIS	KL	0.032	5.120	0.1383	5.254
	LV	**0.028**	**5.075**	**0.2313**	**5.224**
DDS	KL	0.045	5.305	0.1446	6.133
	LV	**0.043**	5.305	**0.1999**	**6.123**

Image density and funnel distribution



Funnel distribution (challenging benchmark for sampling methods):

$$p_{\text{target}}(x) = \mathcal{N}(x_1; 0, 9) \prod_{i=2}^{10} \mathcal{N}(x_i; 0, e^{x_1})$$

Method	Divergence	$\Delta \log \mathcal{Z}(rw) \downarrow$	$\mathcal{W}_\gamma^2 \downarrow$	ESS $\uparrow$	Δstd $\downarrow$
CRAFT		0.123	5.517	-	6.139
PIS	KL	0.111	5.639	**0.1333**	6.921
	LV	**0.097**	**5.593**	0.0746	**6.852**
DIS	KL	0.032	5.120	0.1383	5.254
	LV	**0.028**	**5.075**	**0.2313**	**5.224**
DDS	KL	0.045	5.305	0.1446	6.133
	LV	**0.043**	5.305	**0.1999**	**6.123**

Our improved samplers based on diffusion processes are competitive with state-of-the-art methods based on SMC & normalizing flows (CRAFT).

L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Many-Well problem

Many-Well: typical problem in molecular dynamics with

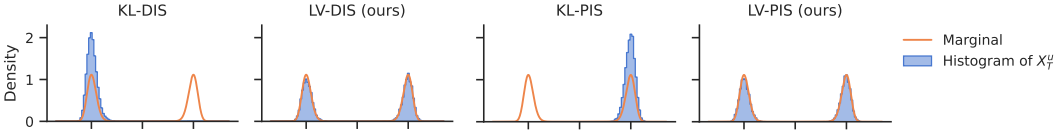
$$\log \rho(x) = -\sum_{i=1}^{w}(x_i^2 - \delta)^2 - \frac{1}{2}\sum_{i=w+1}^{d} x_i^2.$$

L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Many-Well problem

Many-Well: typical problem in molecular dynamics with

$$\log \rho(x) = -\sum_{i=1}^{w}(x_i^2 - \delta)^2 - \frac{1}{2}\sum_{i=w+1}^{d} x_i^2.$$



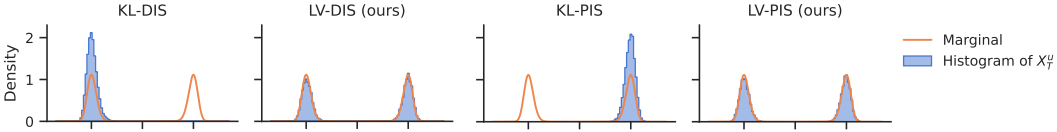
L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Many-Well problem

Many-Well: typical problem in molecular dynamics with

$$\log \rho(x) = -\sum_{i=1}^{w}(x_i^2 - \delta)^2 - \frac{1}{2}\sum_{i=w+1}^{d} x_i^2.$$

Problem	Method	$\Delta \log Z \downarrow$	$\mathcal{W}_\gamma^2 \downarrow$	ESS $\uparrow$	Δ std $\downarrow$
Many-Well	PIS-KL	3.567	1.699	0.0004	1.409
($d = 5, w = 5, \delta = 4$)	**PIS-LV**	**0.214**	**0.121**	**0.6744**	**0.001**
	DIS-KL	1.462	1.175	0.0012	0.431
	DIS-LV	**0.375**	**0.120**	**0.4519**	**0.001**
Many-Well	PIS-KL	0.101	**6.821**	0.8172	0.001
($d = 50, w = 5, \delta = 2$)	**PIS-LV**	**0.087**	6.823	**0.8453**	**0.000**
	DIS-KL	1.785	**6.854**	0.0225	0.009
	DIS-LV	**1.783**	6.855	**0.0227**	0.009



L. Richter and J. Berner. Improved sampling via learned diffusions. In ICLR, 2024

Thank you for your attention!

Website: `https://jberner.info`

Github: `https://github.com/juliusberner`

Mail: `mail@jberner.info`