On Guided & Reversible Solvers for Neural Differential Equations

Zander Whitlock Blasingame

Public defense of the Ph.D. dissertation

Thesis supervisor: Chen Liu

Thesis jury: Mahesh Banavar • Faraz Hussain • Joseph Skufca • Christino Tamon

Clarkson University

2025.04.21

Overview of Contributions Made During the Ph.D. Program

Publications i

1. Z. W. Blasingame and C. Liu (2025a). "A Reversible Solver for Diffusion SDEs". In: *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*. URL: https://openreview.net/forum?id=0gEFLVUL6n
2. Z. W. Blasingame and C. Liu (2025b). "Greed is Good: Guided Generation from a Greedy Perspective". In: *Frontiers in Probabilistic Inference: Learning meets Sampling*. URL: https://openreview.net/forum?id=o4yQzZ5qCW
3. Z. W. Blasingame and C. Liu (2024a). "AdjointDEIS: Efficient Gradients for Diffusion Models". In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., pp. 2449–2483. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/04badd3b048315c8c3a0ca17eff723d7-Paper-Conference.pdf
4. Z. W. Blasingame and C. Liu (2024c). "Greedy-DiM: Greedy Algorithms for Unreasonably Effective Face Morphs". In: *2024 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 1–11. DOI: 10.1109/IJCB62174.2024.10744517

Publications ii

5. Z. W. Blasingame and C. Liu (2024b). "Fast-dim: Towards fast diffusion morphs". In: *IEEE Security & Privacy*
6. Z. W. Blasingame and C. Liu (2024d). "Leveraging diffusion for strong and high quality face morphing attacks". In: *IEEE Transactions on Biometrics, Behavior, and Identity Science* 6.1, pp. 118–131
7. Z. Blasingame, C. Liu, and X. Yao (2021). "Feature Creation Towards the Detection of Non-control-Flow Hijacking Attacks". In: *International Conference on Artificial Neural Networks*. Springer, pp. 153–164
8. Z. Blasingame and C. Liu (2021). "Leveraging adversarial learning for the detection of morphing attacks". In: *2021 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE, pp. 1–8
9. R. E. Neddo, S. Willis, et al. (2025). "Poster: Adapting Pretrained Vision Transformers with LoRA Against Attack Vectors". In: *2025 IEEE International Conference on Mobility, Operations, Services, and Technologies (MOST)*

Publications iii

10. C. Woralert, C. Liu, and Z. Blasingame (2024). "Towards Effective Machine Learning Models for Ransomware Detection via Low-Level Hardware Information". In: *Proceedings of the International Workshop on Hardware and Architectural Support for Security and Privacy 2024*, pp. 10–18
11. R. E. Neddo, Z. W. Blasingame, and C. Liu (2024). "The Impact of Print-Scanning in Heterogeneous Morph Evaluation Scenarios". In: *2024 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 1–10. DOI: `10.1109/IJCB62174.2024.10744441`
12. C. Woralert, C. Liu, Z. Blasingame, and Z. Yang (2023). "A Comparison of One-class and Two-class Models for Ransomware Detection via Low-level Hardware Information". In: *2023 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, pp. 1–6
13. C. Woralert, C. Liu, and Z. Blasingame (2023). "Hard-lite: A lightweight hardware anomaly realtime detection framework targeting ransomware". In: *IEEE Transactions on Circuits and Systems I: Regular Papers*
14. C. Woralert, C. Liu, and Z. Blasingame (2022). "HARD-Lite: A Lightweight Hardware Anomaly Realtime Detection Framework Targeting Ransomware". In: *2022 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, pp. 1–6

15. G. Torres et al. (June 2019). "Detecting Non-Control-Flow Hijacking Attacks Using Contextual Execution Information". In: *Proceedings of the 8th International Workshop on Hardware and Architectural Support for Security and Privacy*. HASP '19. Phœnix, AZ, USA: Association for Computing Machinery. ISBN: 9781450372268. DOI: `10.1145/3337167.3337168`. URL: `https://doi.org/10.1145/3337167.3337168`

Invited Talks

- AdjointDEIS: Efficient Gradients for Diffusion Models — January, 2025
  Mila – Institut Québécois d'intelligence artificielle — *Montréal, Canada*

- Diffusion is all you need for highly effective face morphs — November, 2024
  Transatlantic Dialogue on Presentation Attack Detection — *Washington, D.C.*

- Leveraging Diffusion for Strong and High-Quality Face Morphing Attacks — September, 2024
  IEEE IJCB - Journal Presentation — *Buffalo, NY*

- The Power of Iterative Generative Models for Attacking FR Systems — July, 2024
  Idiap – Institut Dalle Molle d'intelligence artificielle perceptive — *Martigny, Switzerland*

- Diffusion for the Generation of Face Morphs — February, 2024
  CITeR and DSA webinar — *Online*

Introduction

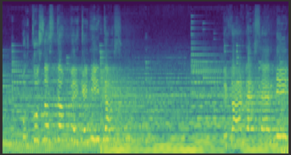
Generative modeling

(a) Image generation (B. 2024)

- Recently, deep generative models have popped off

Generative modeling

(a) Image generation (B. 2024)

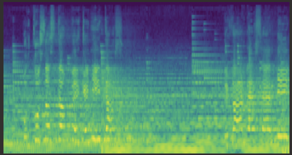


(b) Audio generation (Schneider et al. 2024)

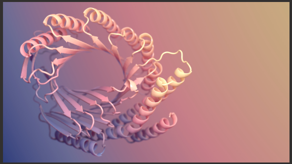
- Recently, deep generative models have popped off

Generative modeling

(a) Image generation (B. 2024)



(b) Audio generation (Schneider et al. 2024)

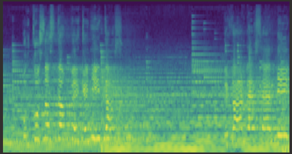


(c) Protein generation (Watson et al. 2023)

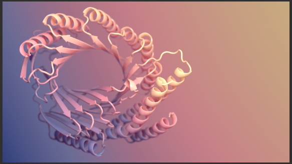
- Recently, deep generative models have popped off

Generative modeling

(a) Image generation (B. 2024)



(b) Audio generation (Schneider et al. 2024)

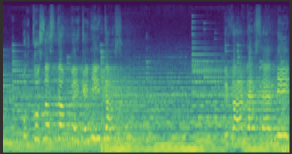


(c) Protein generation (Watson et al. 2023)

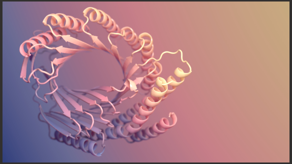
- Recently, deep generative models have popped off
- What is the common factor behind *all* of these models?

Generative modeling

(a) Image generation (B. 2024)



(b) Audio generation (Schneider et al. 2024)



(c) Protein generation (Watson et al. 2023)

- Recently, deep generative models have popped off
- What is the common factor behind *all* of these models?
- They're neural differential equations

What is a neural differential equation anyways?

- Consider some prototypical ODE defined on $[0, T]$, of the form

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t)), \qquad \boldsymbol{x}(t) = \boldsymbol{x}_0 \tag{1}$$

  where $\boldsymbol{f} : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$ is some *nice* vector field

What is a neural differential equation anyways?

- Consider some prototypical ODE defined on $[0, T]$, of the form

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t)), \qquad \boldsymbol{x}(t) = \boldsymbol{x}_0 \tag{1}$$

  where $\boldsymbol{f} : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$ is some *nice* vector field
- A *neural ordinary differential equation* simply replaces the vector field with a neural net

What is a neural differential equation anyways?

- Consider some prototypical ODE defined on $[0, T]$, of the form

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t)), \qquad \boldsymbol{x}(t) = \boldsymbol{x}_0 \tag{1}$$

  where $\boldsymbol{f} : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$ is some *nice* vector field
- A *neural ordinary differential equation* simply replaces the vector field with a neural net
- *I.e.*, we have a neural ODE of the form

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}(t) = \boldsymbol{f}_\theta(t, \boldsymbol{x}(t)), \qquad \boldsymbol{x}(t) = \boldsymbol{x}_0 \tag{2}$$

So how do neural differential equations create?

- Consider $q(\boldsymbol{X}_1)$ which models real-world data

So how do neural differential equations create?

- Consider $q(\boldsymbol{X}_1)$ which models real-world data Call it the target distribution

So how do neural differential equations create?

- Consider $q(\boldsymbol{X}_1)$ which models real-world data
- Consider some tractable $p(\boldsymbol{X}_0)$

So how do neural differential equations create?

- Consider $q(\boldsymbol{X}_1)$ which models real-world data
- Consider some tractable $p(\boldsymbol{X}_0)$ Call it the source distribution

So how do neural differential equations create?

- Consider $q(\boldsymbol{X}_1)$ which models real-world data
- Consider some tractable $p(\boldsymbol{X}_0)$
- Consider the *flow* $\varphi \in C^{1,r}(\mathbb{R} \times \mathbb{R}^d; \mathbb{R}^d)$ which satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t}\varphi_t(\boldsymbol{x}) = \boldsymbol{u}_t(\varphi_t(\boldsymbol{x})) \tag{3}$$

So how do neural differential equations create?

- Consider $q(\boldsymbol{X}_1)$ which models real-world data
- Consider some tractable $p(\boldsymbol{X}_0)$
- Consider the *flow* $\varphi \in C^{1,r}(\mathbb{R} \times \mathbb{R}^d; \mathbb{R}^d)$ which satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t}\varphi_t(\boldsymbol{x}) = \boldsymbol{u}_t(\varphi_t(\boldsymbol{x})) \tag{3}$$

- We want to find a flow (or vector field) such that

$$\varphi_1(\boldsymbol{X}_0) \sim q(\boldsymbol{X}_1) \tag{4}$$

So how do neural differential equations create?

- Consider $q(\boldsymbol{X}_1)$ which models real-world data
- Consider some tractable $p(\boldsymbol{X}_0)$
- Consider the *flow* $\varphi \in C^{1,r}(\mathbb{R} \times \mathbb{R}^d; \mathbb{R}^d)$ which satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t}\varphi_t(\boldsymbol{x}) = \boldsymbol{u}_t(\varphi_t(\boldsymbol{x})) \tag{3}$$

- We want to find a flow (or vector field) such that

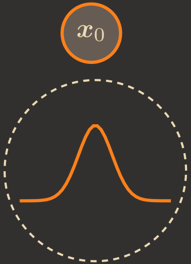
$$\varphi_1(\boldsymbol{X}_0) \sim q(\boldsymbol{X}_1) \tag{4}$$

- Then we aim to find θ such that

$$\boldsymbol{u}_t^{\theta}(\boldsymbol{x}) \approx \boldsymbol{u}_t(\boldsymbol{x}) \tag{5}$$

Sampling with flow models

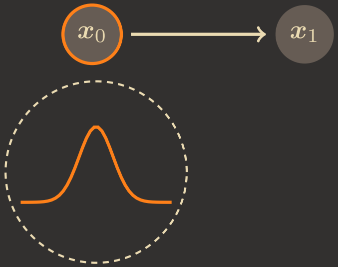
- Consider an Euler scheme with steps $\{t_n\}_{n=0}^{N}$ and step size h



Sampling with flow models

- Consider an Euler scheme with steps $\{t_n\}_{n=0}^{N}$ and step size h
- Now take Euler steps...

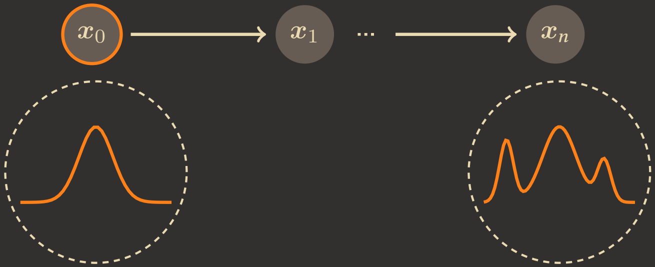
$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + h\boldsymbol{u}_0^{\theta}(\boldsymbol{x}_0)$$



Sampling with flow models

- Consider an Euler scheme with steps $\{t_n\}_{n=0}^{N}$ and step size h
- Now take Euler steps...

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + h\boldsymbol{u}_{n-1}^{\theta}(\boldsymbol{x}_{n-1})$$



Sampling with flow models

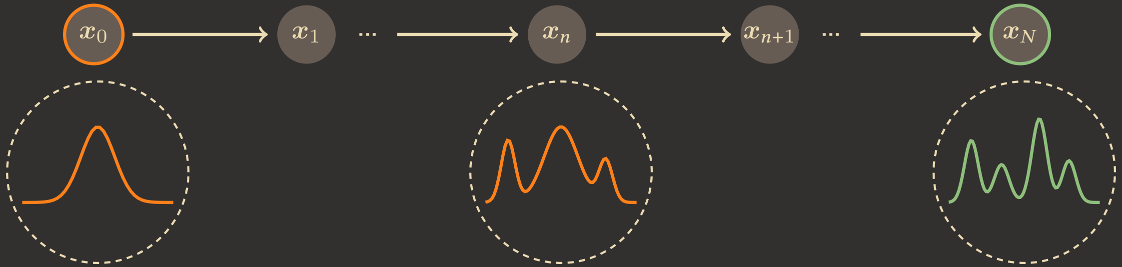
- Consider an Euler scheme with steps $\{t_n\}_{n=0}^{N}$ and step size h
- Now take Euler steps...

$$x_{n+1} = x_n + h u_n^{\theta}(x_n)$$

Sampling with flow models

- Consider an Euler scheme with steps $\{t_n\}_{n=0}^{N}$ and step size h
- Now take Euler steps...

$$\boldsymbol{x}_N = \boldsymbol{x}_{N-1} + h\boldsymbol{u}^{\theta}_{N-1}(\boldsymbol{x}_{N-1})$$



Modern generative paradigm

- Create coupling $(\boldsymbol{X}_0, \boldsymbol{X}_1) \sim p(\boldsymbol{X}_0)q(\boldsymbol{X}_1)$

Modern generative paradigm

- Create coupling $(\boldsymbol{X}_0, \boldsymbol{X}_1) \sim p(\boldsymbol{X}_0)q(\boldsymbol{X}_1)$
- Train flow *(or diffusion)* model to learn $\boldsymbol{u}_t^\theta$

Recap

Modern generative paradigm

- Create coupling $(\boldsymbol{X}_0, \boldsymbol{X}_1) \sim p(\boldsymbol{X}_0)q(\boldsymbol{X}_1)$
- Train flow *(or diffusion)* model to learn $\boldsymbol{u}_t^\theta$
- Sample with numerical ODE *(or SDE)* solver

Recap

Modern generative paradigm

- Create coupling $(\boldsymbol{X}_0, \boldsymbol{X}_1) \sim p(\boldsymbol{X}_0)q(\boldsymbol{X}_1)$
- Train flow *(or diffusion)* model to learn $\boldsymbol{u}_t^\theta$
- Sample with numerical ODE *(or SDE)* solver

Downstream tasks

- Training state-of-the-art flow models is **really expensive** *(1000s of NVIDIA A100 hours)!*

Modern generative paradigm

- Create coupling $(\boldsymbol{X}_0, \boldsymbol{X}_1) \sim p(\boldsymbol{X}_0)q(\boldsymbol{X}_1)$
- Train flow *(or diffusion)* model to learn $\boldsymbol{u}_t^\theta$
- Sample with numerical ODE *(or SDE)* solver

Downstream tasks

- Training state-of-the-art flow models is **really expensive** *(1000s of NVIDIA A100 hours)!*
- Much more efficient to fine-tune or guide $\boldsymbol{u}_t^\theta$ downstream to new tasks

Recap

Modern generative paradigm

- Create coupling $(\boldsymbol{X}_0, \boldsymbol{X}_1) \sim p(\boldsymbol{X}_0)q(\boldsymbol{X}_1)$
- Train flow *(or diffusion)* model to learn $\boldsymbol{u}_t^\theta$
- Sample with numerical ODE *(or SDE)* solver

Downstream tasks

- Training state-of-the-art flow models is **really expensive** *(1000s of NVIDIA A100 hours)!*
- Much more efficient to fine-tune or guide $\boldsymbol{u}_t^\theta$ downstream to new tasks
- Flows are bijective, we can encode samples to edit them – *Latent editing*

Guidance

- Guide model with a function which assesses the output

Problems

Guidance

- Guide model with a function which assesses the output
- Make guidance efficient

Latent editing

Problems

Guidance

- Guide model with a function which assesses the output
- Make guidance efficient

Latent editing

- Perform exact inversion *(algebraic reversibility)*

Problems

Guidance

- Guide model with a function which assesses the output
- Make guidance efficient

Latent editing

- Perform exact inversion *(algebraic reversibility)*
- Make edits resistant to deformations

End-to-end Guidance

Question. How to guide flow/diffusion models?

Z. W. Blasingame and C. Liu (2024a). "AdjointDEIS: Efficient Gradients for Diffusion Models". In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., pp. 2449–2483. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/04badd3b048315c8c3a0ca17eff723d7-Paper-Conference.pdf.

Given $L \in C^1(\mathbb{R}^d; \mathbb{R})$ find $\arg\min_{\boldsymbol{x},\theta} L\left(\varphi_1^\theta(\boldsymbol{x})\right)$

Gradient descent?

Need $\nabla_{\{\boldsymbol{x},\theta\}} L\left(\varphi_1^{\theta}(\boldsymbol{x})\right)$

Discretize-then-optimize (DTO)

- Simplest approach, just backprop through the solver
- Pros: Accuracy of gradients, fast, and easy to implement.
- Cons: Memory intensive $\mathcal{O}(n)$, optimization w.r.t. discretization and not continuous ideal

Discretize-then-optimize (DTO)

- Simplest approach, just backprop through the solver
- Pros: Accuracy of gradients, fast, and easy to implement.
- Cons: Memory intensive $\mathcal{O}(n)$, optimization w.r.t. discretization and not continuous ideal

Optimize-then-discretize (OTD)

- The adjoint method, numerically solve another ODE
- Pros: Memory efficiency $\mathcal{O}(1)$, flexibility
- Cons: Computational cost, truncation errors

Techniques for OTD optimization of flow/diffusion models

Table 1: Overview of different OTD methods for flow/diffusion models.

Method	ODE	SDE	Key Idea
DiffPure (ICML 2022)[1]	✗	✓	First to consider OTD for diffusion models
AdjointDPM (ICLR 2024)[2]	✓	✗	Exponential integrators with OTD
D-Flow (ICML 2024)[3]	✓	✗	OTD with Flow models
AdjointDEIS (NeurIPS 2024)[4]	✓	✓	Bespoke solvers for ODE/SDE, conditioning signals
Implicit Diffusion (AISTATS 2025)[5]	✓	✓	Time parallelization of OTD
OC-Flow (ICLR 2025)[6]	✓	✗	Control flow optimization on Riemannian manifolds

[1] W. Nie et al. (17–23 Jul 2022). "Diffusion Models for Adversarial Purification". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 16805–16827. URL: https://proceedings.mlr.press/v162/nie22a.html.

[2] J. Pan et al. (2024). "AdjointDPM: Adjoint Sensitivity Method for Gradient Backpropagation of Diffusion Probabilistic Models". In: *The Twelfth International Conference on Learning Representations*. URL: https://openreview.net/forum?id=y331DRBgWI.

[3] H. Ben-Hamu et al. (2024). "D-Flow: Differentiating through Flows for Controlled Generation". In: *Forty-first International Conference on Machine Learning*. URL: https://openreview.net/forum?id=SE20BFqj6J.

[4] Z. W. Blasingame and C. Liu (2024a). "AdjointDEIS: Efficient Gradients for Diffusion Models". In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., pp. 2449–2483. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/04badd3b048315c8c3a0ca17eff723d7-Paper-Conference.pdf.

[5] P. Marion et al. (2025). "Implicit Diffusion: Efficient optimization through stochastic sampling". In: *The 28th International Conference on Artificial Intelligence and Statistics*. URL: https://openreview.net/forum?id=r5F7Z8s0Qk.

[6] L. Wang et al. (2025). "Training Free Guided Flow-Matching with Optimal Control". In: *The Thirteenth International Conference on Learning Representations*. URL: https://openreview.net/forum?id=61ss5RA1MM.

Gradients via the continuous adjoint equations

- Pontryagin et al.[7] and later Chen et al.[8] showed how to find these gradients via another ODE

[7] L. S. Pontryagin et al. (1963). "The Mathematical Theory of Optimal Processes.". In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 43.10-11, pp. 514–515. DOI: https://doi.org/10.1002/zamm.19630431023. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/zamm.19630431023. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/zamm.19630431023.

[8] R. T. Chen et al. (2018). "Neural ordinary differential equations". In: *Advances in neural information processing systems* 31.

- Pontryagin et al.[7] and later Chen et al.[8] showed how to find these gradients via another ODE
- Let $\boldsymbol{a}_{\boldsymbol{x}}(t) \coloneqq \partial L/\partial \boldsymbol{x}_t$ and $\boldsymbol{a}_\theta(0) \coloneqq \partial L/\partial \theta$.

[7] L. S. Pontryagin et al. (1963). "The Mathematical Theory of Optimal Processes.". In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 43.10-11, pp. 514–515. DOI: https://doi.org/10.1002/zamm.19630431023. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/zamm.19630431023. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/zamm.19630431023.

[8] R. T. Chen et al. (2018). "Neural ordinary differential equations". In: *Advances in neural information processing systems* 31.

Gradients via the continuous adjoint equations

- Pontryagin et al.[7] and later Chen et al.[8] showed how to find these gradients via another ODE
- Let $\boldsymbol{a}_{\boldsymbol{x}}(t) := \partial L / \partial \boldsymbol{x}_t$ and $\boldsymbol{a}_\theta(0) := \partial L / \partial \theta$.
- Then we have the *continuous adjoint equations*

$$\begin{aligned} \boldsymbol{a}_{\boldsymbol{x}}(1) &= \frac{\partial L}{\partial \boldsymbol{x}_1}, & \frac{\mathrm{d}\boldsymbol{a}_{\boldsymbol{x}}}{\mathrm{d}t}(t) &= -\boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{u}_t^\theta}{\partial \boldsymbol{x}}(\boldsymbol{x}_t), \\ \boldsymbol{a}_\theta(1) &= 0, & \frac{\mathrm{d}\boldsymbol{a}_\theta}{\mathrm{d}t}(t) &= -\boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{u}_t^\theta}{\partial \theta}(\boldsymbol{x}_t). \end{aligned} \tag{6}$$

7 L. S. Pontryagin et al. (1963). "The Mathematical Theory of Optimal Processes.". In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 43.10-11, pp. 514–515. DOI: `https://doi.org/10.1002/zamm.19630431023`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/zamm.19630431023`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/zamm.19630431023`.

8 R. T. Chen et al. (2018). "Neural ordinary differential equations". In: *Advances in neural information processing systems* 31.

Affine conditional flows

- Consider the flow model admitted by

$$\boldsymbol{X}_t = \alpha_t \boldsymbol{X}_1 + \sigma_t \boldsymbol{X}_0 \tag{7}$$

Affine conditional flows

- Consider the flow model admitted by

$$\boldsymbol{X}_t = \alpha_t \boldsymbol{X}_1 + \sigma_t \boldsymbol{X}_0 \tag{7}$$

- Called an *affine conditional flow*

Affine conditional flows

- Consider the flow model admitted by

$$\boldsymbol{X}_t = \alpha_t \boldsymbol{X}_1 + \sigma_t \boldsymbol{X}_0 \tag{7}$$

- Called an *affine conditional flow*
- Vector field is given by

$$\begin{aligned} \boldsymbol{u}_t(\boldsymbol{x}) &= \frac{\dot{\sigma}_t}{\sigma_t}\boldsymbol{x} + \left[\dot{\alpha}_t - \alpha_t \frac{\dot{\sigma}_t}{\sigma_t}\right] \mathbb{E}[\boldsymbol{X}_1 | \boldsymbol{X}_t = \boldsymbol{x}] & (8) \\ &= \frac{\dot{\alpha}_t}{\alpha_t}\boldsymbol{x} + \left[\dot{\sigma}_t - \sigma_t \frac{\dot{\alpha}_t}{\alpha_t}\right] \mathbb{E}[\boldsymbol{X}_0 | \boldsymbol{X}_t = \boldsymbol{x}] & (9) \end{aligned}$$

Affine conditional flows

- Consider the flow model admitted by

$$\boldsymbol{X}_t = \alpha_t \boldsymbol{X}_1 + \sigma_t \boldsymbol{X}_0 \tag{7}$$

- Called an *affine conditional flow*
- Vector field is given by

$$\boldsymbol{u}_t(\boldsymbol{x}) = \frac{\dot{\sigma}_t}{\sigma_t}\boldsymbol{x} + \left[\dot{\alpha}_t - \alpha_t \frac{\dot{\sigma}_t}{\sigma_t}\right] \boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x}) \tag{8}$$

$$= \frac{\dot{\alpha}_t}{\alpha_t}\boldsymbol{x} + \left[\dot{\sigma}_t - \sigma_t \frac{\dot{\alpha}_t}{\alpha_t}\right] \boldsymbol{x}^{\theta}_{0|t}(\boldsymbol{x}) \tag{9}$$

Proposition 1 (Continuous adjoint equations for target and source prediction models). Given an affine conditional flow model with learnt vector field $\boldsymbol{u}_t^\theta$, then the continuous adjoint equations in Eq. (6) can be rewritten w.r.t. to the target prediction model and the source prediction model as

$$\frac{\mathrm{d}\boldsymbol{a}_{\boldsymbol{x}}}{\mathrm{d}t}(t) = -\frac{\dot{\sigma}_t}{\sigma_t}\boldsymbol{a}_{\boldsymbol{x}}(t) - \left[\dot{\alpha}_t - \alpha_t\frac{\dot{\sigma}_t}{\sigma_t}\right]\boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{x}_{1|t}^\theta}{\partial \boldsymbol{x}_t}(\boldsymbol{x}_t), \tag{10}$$

$$= -\frac{\dot{\alpha}_t}{\alpha_t}\boldsymbol{a}_{\boldsymbol{x}}(t) - \left[\dot{\sigma}_t - \sigma_t\frac{\dot{\alpha}_t}{\alpha_t}\right]\boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{x}_{0|t}^\theta}{\partial \boldsymbol{x}_t}(\boldsymbol{x}_t); \tag{11}$$

likewise, the solvers for $\boldsymbol{a}_\theta$ can be found, *mutatis mutandis*, to be

$$\frac{\mathrm{d}\boldsymbol{a}_\theta}{\mathrm{d}t}(t) = -\left[\dot{\alpha}_t - \alpha_t\frac{\dot{\sigma}_t}{\sigma_t}\right]\boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{x}_{1|t}^\theta}{\partial \theta}(\boldsymbol{x}_t), \tag{12}$$

$$= -\left[\dot{\sigma}_t - \sigma_t\frac{\dot{\alpha}_t}{\alpha_t}\right]\boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{x}_{0|t}^\theta}{\partial \theta}(\boldsymbol{x}_t). \tag{13}$$

Proposition 2 (Exact solution of the continuous adjoint equations for source prediction models). Given an initial value $[\boldsymbol{a}_{\boldsymbol{x}}(t), \boldsymbol{a}_\theta(t)]$ at time $t \in (0,1]$, the solution $[\boldsymbol{a}_{\boldsymbol{x}}(s), \boldsymbol{a}_\theta(s)]$ at time $s \in [0,t)$ to the continuous adjoint equations for source prediction models described in Proposition 1 is given by

$$
\begin{aligned}
\boldsymbol{a}_{\boldsymbol{x}}(s) &= \frac{\alpha_t}{\alpha_s}\boldsymbol{a}_{\boldsymbol{x}}(t) + \frac{1}{\alpha_s}\int_t^s \alpha_\lambda^2 e^{-\lambda}\boldsymbol{a}_{\boldsymbol{x}}(\lambda)^\top \frac{\partial \boldsymbol{x}_{0|\lambda}^\theta}{\partial \boldsymbol{x}_\lambda}(\boldsymbol{x}_\lambda)\,\mathrm{d}\lambda,\\
\boldsymbol{a}_\theta(s) &= \boldsymbol{a}_\theta(t) + \int_t^s \alpha_\lambda e^{-\lambda}\boldsymbol{a}_{\boldsymbol{x}}(\lambda)^\top \frac{\partial \boldsymbol{x}_{0|\lambda}^\theta}{\partial \boldsymbol{x}_\lambda}(\boldsymbol{x}_\lambda)\,\mathrm{d}\lambda,
\end{aligned}
\tag{14}
$$

where $\lambda_t \coloneqq \log \alpha_t/\sigma_t$.

- We denote the *scaled* vector-Jacobian product by

$$\boldsymbol{V}(\boldsymbol{x};t) = \alpha_t^2 \boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{x}_{0|t}^\theta}{\partial \boldsymbol{x}_t}(\boldsymbol{x}_t) \tag{15}$$

Designing bespoke ODE solvers

- We denote the *scaled* vector-Jacobian product by

$$\boldsymbol{V}(\boldsymbol{x};t) = \alpha_t^2 \boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{x}_{0|t}^{\theta}}{\partial \boldsymbol{x}_t}(\boldsymbol{x}_t) \tag{15}$$

- Use Taylor Expansion on Eq. (14)

Designing bespoke ODE solvers

- We denote the *scaled* vector-Jacobian product by

$$\boldsymbol{V}(x;t) = \alpha_t^2 \boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{x}_{0|t}^{\theta}}{\partial \boldsymbol{x}_t}(\boldsymbol{x}_t) \tag{15}$$

- Use Taylor Expansion on Eq. (14)
- Let $\boldsymbol{V}^{(n)}$ denote the n-th derivative w.r.t. λ

- We denote the *scaled* vector-Jacobian product by

$$V(x;t) = \alpha_t^2 \boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{x}_{0|t}^{\theta}}{\partial \boldsymbol{x}_t}(\boldsymbol{x}_t) \tag{15}$$

- Use Taylor Expansion on Eq. (14)
- Let $\boldsymbol{V}^{(n)}$ denote the n-th derivative w.r.t. λ
- Let $h = \lambda_s - \lambda_t$,

$$\boldsymbol{a}_{\boldsymbol{x}}(s) = \underbrace{\frac{\alpha_t}{\alpha_s}\boldsymbol{a}_{\boldsymbol{x}}(t)}_{\substack{\text{Linear term}\\ \textbf{Exactly computed}}} + \frac{1}{\alpha_s}\sum_{n=0}^{k-1} + \tag{16}$$

- We denote the *scaled* vector-Jacobian product by

$$\boldsymbol{V}(\boldsymbol{x};t) = \alpha_t^2 \boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{x}_{0|t}^\theta}{\partial \boldsymbol{x}_t}(\boldsymbol{x}_t) \tag{15}$$

- Use Taylor Expansion on Eq. (14)
- Let $\boldsymbol{V}^{(n)}$ denote the n-th derivative w.r.t. λ
- Let $h = \lambda_s - \lambda_t$,

$$\boldsymbol{a}_{\boldsymbol{x}}(s) = \underbrace{\frac{\alpha_t}{\alpha_s}\boldsymbol{a}_{\boldsymbol{x}}(t)}_{\substack{\text{Linear term} \\ \textbf{Exactly computed}}} + \frac{1}{\alpha_s}\sum_{n=0}^{k-1} \underbrace{\boldsymbol{V}^{(n)}(\boldsymbol{x};\lambda_t)}_{\substack{\text{Derivatives} \\ \textbf{Approximated}}} + \tag{16}$$

Designing bespoke ODE solvers

- We denote the *scaled* vector-Jacobian product by

$$\boldsymbol{V}(\boldsymbol{x};t) = \alpha_t^2 \boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{x}_{0|t}^{\theta}}{\partial \boldsymbol{x}_t}(\boldsymbol{x}_t) \tag{15}$$

- Use Taylor Expansion on Eq. (14)
- Let $\boldsymbol{V}^{(n)}$ denote the n-th derivative w.r.t. λ
- Let $h = \lambda_s - \lambda_t$,

$$\boldsymbol{a}_{\boldsymbol{x}}(s) = \underbrace{\frac{\alpha_t}{\alpha_s}\boldsymbol{a}_{\boldsymbol{x}}(t)}_{\substack{\text{Linear term} \\ \textbf{Exactly computed}}} + \frac{1}{\alpha_s}\sum_{n=0}^{k-1} \underbrace{\boldsymbol{V}^{(n)}(\boldsymbol{x};\lambda_t)}_{\substack{\text{Derivatives} \\ \textbf{Approximated}}} \underbrace{\int_{\lambda_t}^{\lambda_s} \frac{(\lambda-\lambda_t)^n}{n!} e^{-\lambda}\,\mathrm{d}\lambda}_{\substack{\text{Cœfficients} \\ \textbf{Analytically computed}}} + \tag{16}$$

- We denote the *scaled* vector-Jacobian product by

$$\boldsymbol{V}(\boldsymbol{x}; t) = \alpha_t^2 \boldsymbol{a}_{\boldsymbol{x}}(t)^\top \frac{\partial \boldsymbol{x}_{0|t}^\theta}{\partial \boldsymbol{x}_t}(\boldsymbol{x}_t) \tag{15}$$

- Use Taylor Expansion on Eq. (14)
- Let $\boldsymbol{V}^{(n)}$ denote the n-th derivative w.r.t. λ
- Let $h = \lambda_s - \lambda_t$,

$$\boldsymbol{a}_{\boldsymbol{x}}(s) = \underbrace{\frac{\alpha_t}{\alpha_s}\boldsymbol{a}_{\boldsymbol{x}}(t)}_{\substack{\text{Linear term} \\ \textbf{Exactly computed}}} + \frac{1}{\alpha_s}\sum_{n=0}^{k-1} \underbrace{\boldsymbol{V}^{(n)}(\boldsymbol{x}; \lambda_t)}_{\substack{\text{Derivatives} \\ \textbf{Approximated}}} \underbrace{\int_{\lambda_t}^{\lambda_s} \frac{(\lambda - \lambda_t)^n}{n!} e^{-\lambda} \, \mathrm{d}\lambda}_{\substack{\text{Cœfficients} \\ \textbf{Analytically computed}}} + \underbrace{\mathcal{O}(h^{k+1})}_{\substack{\text{Higher-order errors} \\ \textbf{Omitted}}} \tag{16}$$

First-order solver

Given an initial augmented adjoint state $[\boldsymbol{a}_{\boldsymbol{x}}(t), \boldsymbol{a}_{\theta}(t)]$ at time $t \in (0, 1]$, the solution $[\boldsymbol{a}_{\boldsymbol{x}}(s), \boldsymbol{a}_{\theta}(s)]$ at time $s \in [0, t)$ is approximated by

$$\begin{aligned}
\boldsymbol{a}_{\boldsymbol{x}}(s) &= \frac{\alpha_t}{\alpha_s}\boldsymbol{a}_{\boldsymbol{x}}(t) + \frac{\sigma_s}{\alpha_s^2}(e^h - 1)\boldsymbol{V}(\boldsymbol{x}; t)\\
\boldsymbol{a}_{\theta}(s) &= \boldsymbol{a}_{\theta}(t) + \frac{\sigma_s}{\alpha_s}(e^h - 1)\boldsymbol{V}(\theta; t).
\end{aligned} \tag{17}$$

Recap

- Guide generative process with L

Recap

- Guide generative process with L
- Backprop though discretization – high memory consumption $\mathcal{O}(n)$

Recap

- Guide generative process with L
- Backprop though discretization – high memory consumption $\mathcal{O}(n)$
- Solve another ODE which models gradients – constant memory $\mathcal{O}(1)$

Recap

- Guide generative process with L
- Backprop though discretization – high memory consumption $\mathcal{O}(n)$
- Solve another ODE which models gradients – constant memory $\mathcal{O}(1)$
- Lots of compute

Recap

- Guide generative process with L
- Backprop though discretization – high memory consumption $\mathcal{O}(n)$
- Solve another ODE which models gradients – constant memory $\mathcal{O}(1)$
- Lots of compute
- Backwards solve is *decoupled*

Recap

- Guide generative process with L
- Backprop though discretization – high memory consumption $\mathcal{O}(n)$
- Solve another ODE which models gradients – constant memory $\mathcal{O}(1)$
- Lots of compute
- Backwards solve is *decoupled*
- What's the smallest number of discretization steps we can get away with?

Greedy Guidance

Preface to the section

Question. How to *efficiently* compute the gradients?

Z. W. Blasingame and C. Liu (2025b). "Greed is Good: Guided Generation from a Greedy Perspective". In: *Frontiers in Probabilistic Inference: Learning meets Sampling*. URL: https://openreview.net/forum?id=o4yQzZ5qCW.

Posterior sampling

- Recall the target prediction formula

$$\boldsymbol{x}_{1|t}(\boldsymbol{x}) = \mathbb{E}[\boldsymbol{X}_1|\boldsymbol{X}_t = \boldsymbol{x}] \tag{18}$$

Posterior sampling

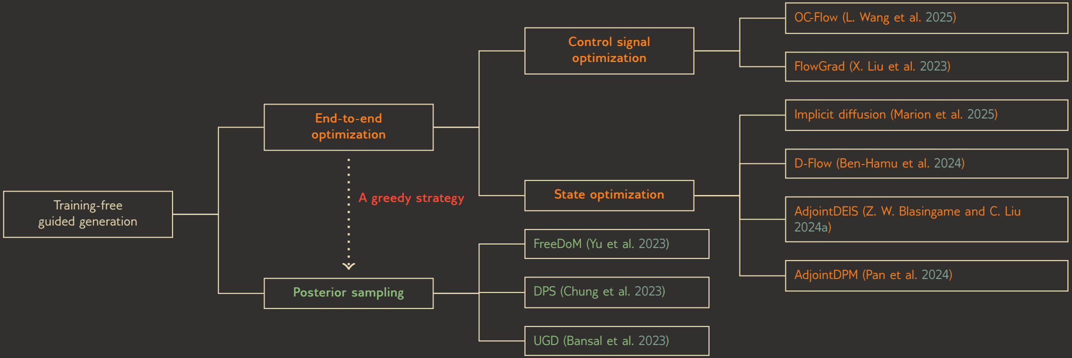
- Recall the target prediction formula

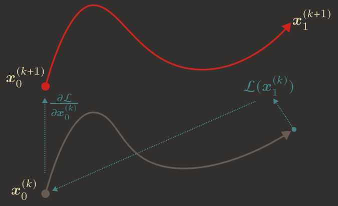
$$\boldsymbol{x}_{1|t}(\boldsymbol{x}) = \mathbb{E}[\boldsymbol{X}_1 | \boldsymbol{X}_t = \boldsymbol{x}] \tag{18}$$

- Use this estimate of the posterior to perform guidance,

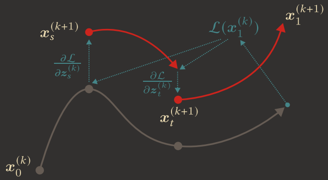
$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}_{1|t}(\boldsymbol{x})) \tag{19}$$

We view posterior guidance as a *greedy strategy* of end-to-end guidance


Training-free guided generation
End-to-end optimization
A greedy strategy
Posterior sampling
Control signal optimization
State optimization
FreeDoM (Yu et al. 2023)
DPS (Chung et al. 2023)
UGD (Bansal et al. 2023)
OC-Flow (L. Wang et al. 2025)
FlowGrad (X. Liu et al. 2023)
Implicit diffusion (Marion et al. 2025)
D-Flow (Ben-Hamu et al. 2024)
AdjointDEIS (Z. W. Blasingame and C. Liu 2024a)
AdjointDPM (Pan et al. 2024)

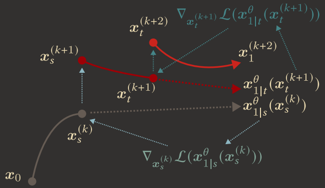


State optimization



Control signal optimization

End-to-end guidance



Posterior guidance

Proposition 3 (Exact solution of affine probability paths). Given an initial value of $\boldsymbol{x}_s$ at time $s \in [0, 1]$ the solution $\boldsymbol{x}_t$ at time $t \in [0, 1]$ of an affine probability path is:

$$\boldsymbol{x}_t = \frac{\sigma_t}{\sigma_s}\boldsymbol{x}_s + \sigma_t \int_{\gamma_s}^{\gamma_t} \boldsymbol{x}^{\theta}_{1|\gamma}(\boldsymbol{x}_\gamma)\, \mathrm{d}\gamma, \tag{20}$$

where $\gamma_t = \alpha_t/\sigma_t$.

Greedy as an explicit Euler scheme with DTO

- Consider the Taylor expansion of Eq. (20)

$$\boldsymbol{x}_t = \frac{\sigma_t}{\sigma_s}\boldsymbol{x}_s + \sigma_t \sum_{n=0}^{k-1} \frac{\mathrm{d}^n}{\mathrm{d}\gamma^n}\left[\boldsymbol{x}^{\theta}_{1|\gamma}(\boldsymbol{x}_\gamma)\right]_{\gamma=\gamma_s} \frac{h^{n+1}}{n!} + \mathcal{O}(h^{k+1}) \tag{21}$$

Greedy as an explicit Euler scheme with DTO

- Consider the Taylor expansion of Eq. (20)
- Then, the first-order expansion is

$$x_t = \frac{\sigma_t}{\sigma_s} x_s + \sigma_t h x^{\theta}_{1|s}(x_s) + \mathcal{O}(h) \tag{21}$$

Greedy as an explicit Euler scheme with DTO

- Consider the Taylor expansion of Eq. (20)
- Then, the first-order expansion is
- Drop high-order error terms

$$\boldsymbol{x}_t \approx \frac{\sigma_t}{\sigma_s}\boldsymbol{x}_s + \sigma_t h \boldsymbol{x}^{\theta}_{1|s}(\boldsymbol{x}_s) \tag{21}$$

Greedy as an explicit Euler scheme with DTO

- Consider the Taylor expansion of Eq. (20)
- Then, the first-order expansion is
- Drop high-order error terms

$$\boldsymbol{x}_t \approx \frac{\sigma_t}{\sigma_s}\boldsymbol{x}_s + \sigma_t \left(\frac{\alpha_t}{\sigma_t} - \frac{\alpha_s}{\sigma_s}\right)\boldsymbol{x}^{\theta}_{1|s}(\boldsymbol{x}_s) \tag{21}$$

Greedy as an explicit Euler scheme with DTO

- Consider the Taylor expansion of Eq. (20)
- Then, the first-order expansion is
- Drop high-order error terms

$$\boldsymbol{x}_t \approx \frac{\sigma_t}{\sigma_s}\boldsymbol{x}_s + \left(\alpha_t - \alpha_s\frac{\sigma_t}{\sigma_s}\right)\boldsymbol{x}^{\theta}_{1|s}(\boldsymbol{x}_s) \tag{21}$$

Greedy as an explicit Euler scheme with DTO

- Consider the Taylor expansion of Eq. (20)
- Then, the first-order expansion is
- Drop high-order error terms
- In the limit as $t \to 1$, $\alpha_t \to 1$, $\sigma_t \to 0$

$$\boldsymbol{x}_1 \approx \boldsymbol{x}^{\theta}_{1|s}(\boldsymbol{x}_s) \tag{21}$$

Greedy as an explicit Euler scheme with DTO

- Consider the Taylor expansion of Eq. (20)
- Then, the first-order expansion is
- Drop high-order error terms
- In the limit as $t \to 1$, $\alpha_t \to 1$, $\sigma_t \to 0$

$$\boldsymbol{x}_1 \approx \boldsymbol{x}^{\theta}_{1|s}(\boldsymbol{x}_s) \tag{21}$$

- Hence, the greedy gradient $\nabla_{\boldsymbol{x}} L(\boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x}_t))$, can be viewed as a DTO scheme with a large explicit Euler step.

Greedy as an implicit Euler scheme with OTD

- Now consider an OTD scheme

Greedy as an implicit Euler scheme with OTD

- Now consider an OTD scheme
- Continuous adjoint equations have a form of

$$\boldsymbol{a}_{\boldsymbol{x}}(s) = \frac{\sigma_t}{\sigma_s}\boldsymbol{a}_{\boldsymbol{x}}(t) + \sigma_t \int_{\gamma_s}^{\gamma_t} \boldsymbol{a}_{\boldsymbol{x}}(\gamma)^\top \frac{\partial \boldsymbol{x}_{1|\gamma}^{\theta}(\boldsymbol{x}_\gamma)}{\partial \boldsymbol{x}_\gamma}\,\mathrm{d}\gamma \tag{22}$$

Greedy as an implicit Euler scheme with OTD

- Now consider an OTD scheme
- Continuous adjoint equations have a form of

$$\boldsymbol{a}_{\boldsymbol{x}}(s) = \frac{\sigma_t}{\sigma_s}\boldsymbol{a}_{\boldsymbol{x}}(t) + \sigma_t \int_{\gamma_s}^{\gamma_t} \boldsymbol{a}_{\boldsymbol{x}}(\gamma)^\top \frac{\partial \boldsymbol{x}_{1|\gamma}^\theta(\boldsymbol{x}_\gamma)}{\partial \boldsymbol{x}_\gamma} \, \mathrm{d}\gamma \tag{22}$$

- Then in the limit as $t \to 1$ the first iteration of a fixed-point iteration scheme yields

$$\boldsymbol{a}_{\boldsymbol{x}}(s) \approx \boldsymbol{a}_{\boldsymbol{x}}(1)^\top \frac{\partial \boldsymbol{x}_{1|t}^\theta(\boldsymbol{x}_s)}{\partial \boldsymbol{x}_s} = \nabla_{\boldsymbol{x}} L(\boldsymbol{x}_{1|s}^\theta(\boldsymbol{x}_s)) \tag{23}$$

(a) Identity a **(b)** Face morphing with DiM **(c)** Identity b

- Z. W. Blasingame and C. Liu (2024d) proposed Diffusion Morphs (DiM)
- We apply a greedy gradient strategy to DiM and compare it to end-to-end optimization with DiM

Experimental case study – Face morphing

(a) Identity a **(b)** Webmorph **(c)** DiM **(d)** MIPGAN-II **(e)** Identity b

Comparison of different face morphing pipelines on the FRLL (DeBruine and Jones 2017) dataset

Experimental case study – Face morphing

(a) Identity a **(b)** DiM-A **(c)** Fast-DiM **(d)** Morph-PIPE **(e)** Adjoint-DiM **(f)** Greedy-DiM **(g)** Identity b

Comparison of ODE-based DiM morphs on the FRLL (DeBruine and Jones 2017) dataset

Experimental case study – Face morphing

Morphing Attack	Framework	NFE(↓)	MMPMR @ FMR = 0.1%(↑) AdaFace	ArcFace	ElasticFace
FaceMorpher (Huber et al. 2022)	Landmark	-	89.78	87.73	89.57
Webmorph (Huber et al. 2022)	Landmark	-	**97.96**	**96.93**	**98.36**
OpenCV (Huber et al. 2022)	Landmark	-	94.48	92.43	94.27
MIPGAN-I (H. Zhang, Venkatesh, et al. 2021)	GAN	-	72.19	77.51	66.46
MIPGAN-II (H. Zhang, Venkatesh, et al. 2021)	GAN	-	70.55	72.19	65.24
DiM-A (Z. W. Blasingame and C. Liu 2024d)	DiM	350	92.23	90.18	93.05
Fast-DiM (Z. W. Blasingame and C. Liu 2024b)	DiM	300	92.02	90.18	93.05
Morph-PIPE (H. Zhang, Ramachandra, et al. 2023)	DiM	2350	95.91	92.84	95.5
Adjoint-DiM (Z. W. Blasingame and C. Liu 2024a)	DiM	2250	**99.8**	**98.77**	**99.39**
Greedy-DiM (Z. W. Blasingame and C. Liu 2024c)	DiM	270	100	100	100

Effectiveness of face morphing attacks on the SYN-MAD 2022 (Huber et al. 2022) evaluation dataset

1st place, **2nd place**, **3rd place**

Proposition 4 (Gradient of target prediction model (Ben-Hamu et al. 2024)). For affine Gaussian probability paths, the gradient of the target prediction model $\boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x})$ w.r.t. $\boldsymbol{x}$ is proportional to the variance of $p_{1|t}(\boldsymbol{x}_1|\boldsymbol{x})$, *i.e.*,

$$\nabla_{\boldsymbol{x}} \boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x}) = \frac{\alpha_t}{\sigma_t^2} \mathrm{Var}_{1|t}(\boldsymbol{x}), \tag{24}$$

where

$$\mathrm{Var}_{1|t}(\boldsymbol{x}) = \mathbb{E}_{p_{1|t}(\boldsymbol{x}_1|\boldsymbol{x})} \left[(\boldsymbol{x}_1 - \boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x}))(\boldsymbol{x}_1 - \boldsymbol{x}^{\theta}_{1|t}(\boldsymbol{x}))^{\top} \right]. \tag{25}$$

Theorem 5 ((Ben-Hamu et al. 2024)). For the standard affine Gaussian probability path, the differential of $\varphi^{\theta}_{0,1}(\boldsymbol{x})$ as of function of $\boldsymbol{x}$ is

$$\nabla_{\boldsymbol{x}} \varphi^{\theta}_{0,1}(\boldsymbol{x}) = \sigma_1 \mathcal{T} \exp \left[\int_0^1 \frac{1}{2} \dot{\gamma}_t \operatorname{Var}_{1|t}(\boldsymbol{x}) \, \mathrm{d}t \right], \tag{26}$$

where $\mathcal{T}\exp$ denotes the time-ordered exponential.

The time-ordered exponential is defined as

$$\mathcal{T} \exp \left[\int_0^t \boldsymbol{A}(s) \, \mathrm{d}s \right] = \sum_{n=1}^{\infty} \frac{(-1)^n}{n!} \int_0^t \mathrm{d}s_1 \cdots \int_0^t \mathrm{d}s_n \;\; \mathcal{T}\{\boldsymbol{A}(s_1) \dots \boldsymbol{A}(s_n)\} \tag{27}$$

Proposition 6 (Dynamics of greedy gradient guidance). Consider the standard affine Gaussian probability paths model trained to zero loss. The Gateaux differential of $\boldsymbol{x}$ at some time $t \in [0,1]$ in the direction of the gradient $\nabla_{\boldsymbol{x}} \mathcal{L}\left(\boldsymbol{x}_{1|t}^{\theta}(\boldsymbol{x})\right)$ is given by

$$\delta_{\boldsymbol{x}}^{\mathcal{G}} \varphi_{t,1}^{\theta}(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} \varphi_{t,1}^{\theta}(\boldsymbol{x}) \nabla_{\boldsymbol{x}} \boldsymbol{x}_{1|t}^{\theta}(\boldsymbol{x})^{\top} \nabla_{\boldsymbol{x}_1} \mathcal{L}(\boldsymbol{x}_1). \tag{28}$$

Reversible Solvers

Preface to the section

Question 1. How to perform exact inversion with diffusion models?
Question 2. Whilst minimizing distortions of edits?

Z. W. Blasingame and C. Liu (2025a). "A Reversible Solver for Diffusion SDEs". In: *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*. URL: `https://openreview.net/forum?id=0gEFLVUL6n`

Exact inversion with flow/diffusion models

- Flows, by definition, are bijective

Exact inversion with flow/diffusion models

- Flows, by definition, are bijective
- Then

$$x = (\varphi_1 \circ \varphi_1^{-1})(x) \tag{29}$$

Exact inversion with flow/diffusion models

- Flows, by definition, are bijective
- Then

$$x = (\varphi_1 \circ \varphi_1^{-1})(x) \tag{29}$$

- But in practice we use a numerical solver

Exact inversion with flow/diffusion models

- Flows, by definition, are bijective
- Then

$$x = (\varphi_1 \circ \varphi_1^{-1})(x) \tag{29}$$

- But in practice we use a numerical solver
- Thus there will *likely* be a misalignment of the truncation errors

Exact inversion with flow/diffusion models

- Flows, by definition, are bijective
- Then

$$x = (\varphi_1 \circ \varphi_1^{-1})(x) \tag{29}$$

- But in practice we use a numerical solver
- Thus there will *likely* be a misalignment of the truncation errors
- To solve this we need *algebraically reversible* numerical solvers

Algebraically reversible solvers

- Consider a numerical scheme

$$(\boldsymbol{x}_n, \boldsymbol{\alpha}_n) \mapsto (\boldsymbol{x}_{n+1}, \boldsymbol{\alpha}_{n+1}), \tag{30}$$

where $\boldsymbol{\alpha}_n$ denotes auxiliary information stored by the scheme

Algebraically reversible solvers

- Consider a numerical scheme

$$(\boldsymbol{x}_n, \boldsymbol{\alpha}_n) \mapsto (\boldsymbol{x}_{n+1}, \boldsymbol{\alpha}_{n+1}), \tag{30}$$

  where α_n denotes auxiliary information stored by the scheme
- Such a solver is *algebraically reversible* if

$$(\boldsymbol{x}_{n+1}, \boldsymbol{\alpha}_{n+1}) \mapsto (\boldsymbol{x}_n, \boldsymbol{\alpha}_n), \tag{31}$$

  can be written in closed form

Overview of reversible solvers

Solver	Number of extra states	Local truncation error	Region of linear stability	Proof of convergence
General ODEs				
Asynchronous leapfrog (ICLR 2021)[9]	1	$\mathcal{O}(h^3)$	✗	✓
Reversible Heun (NeurIPS 2021)[10]	1	$\mathcal{O}(h^3)$	✗	✓
McCallum-Foster (Pre-print 2024)[11]	1	$\mathcal{O}(h^{k+1})$	✓	✓
Probability flow ODEs				
EDICT (CVPR 2023)[12]	1	$\mathcal{O}(h)$	✗	✗
BDIA (ECCV 2024)[13]	1	$\mathcal{O}(h^2)$	✗	✗
BELM (NeurIPS 2024)[14]	$k-1$	$\mathcal{O}(h^{k+1})$	✗	$\sim$

[9] J. Zhuang et al. (2021). “MALI: A memory efficient and reverse accurate integrator for Neural ODEs”. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=blfSjHeFM_e.

[10] P. Kidger et al. (2021). “Efficient and accurate gradients for neural sdes”. In: *Advances in Neural Information Processing Systems* 34, pp. 18747–18761.

[11] S. McCallum and J. Foster (2024). “Efficient, Accurate and Stable Gradients for Neural ODEs”. In: *arXiv preprint arXiv:2410.11648*.

[12] B. Wallace, A. Gokul, and N. Naik (2023). “Edict: Exact diffusion inversion via coupled transformations”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22532–22541.

[13] G. Zhang, J. P. Lewis, and W. B. Kleijn (2024). “Exact Diffusion Inversion via Bidirectional Integration Approximation”. In: *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LVII*. Milan, Italy: Springer-Verlag, pp. 19–36. ISBN: 978-3-031-72997-3. DOI: 10.1007/978-3-031-72998-0_2. URL: https://doi.org/10.1007/978-3-031-72998-0_2.

[14] F. Wang et al. (2024). “BELM: Bidirectional Explicit Linear Multi-step Sampler for Exact Inversion in Diffusion Models”. In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. URL: https://openreview.net/forum?id=ccQ4fmwLDb.

Minimizing distortion of edits

Figure 5: Change from *lion* to *tiger*. From left to right: Source image, SDE Solve, ODE Solve. From S. Nie et al. (2024)

SDEs contract errors, ODEs are preserve errors[15]

[15] S. Nie et al. (2024). "The Blessing of Randomness: SDE Beats ODE in General Diffusion-based Image Editing". In: *The Twelfth International Conference on Learning Representations*. URL: `https://openreview.net/forum?id=DesYwmUG00`.

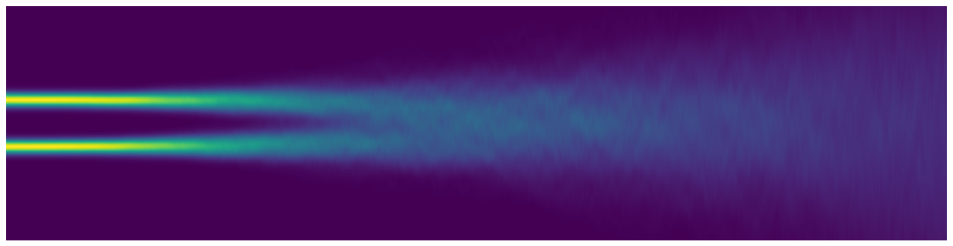
Minimizing distortion of edits

(a) Identity a **(b)** DiM (ODE Solver) **(c)** DiM (SDE Solver) **(d)** Identity b

Diffusion models as SDEs

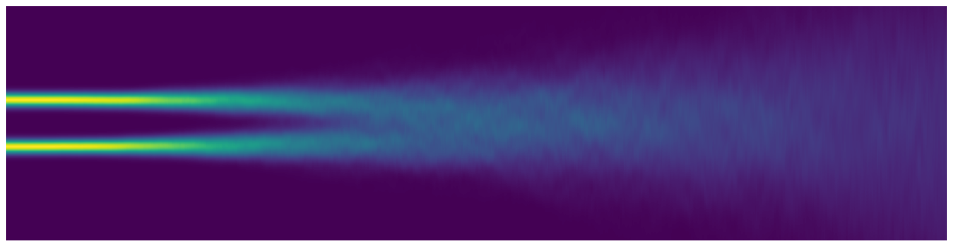
$$\mathrm{d}\boldsymbol{X}_t = \boldsymbol{f}(\boldsymbol{X}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{W}_t$$

$\boldsymbol{x}_0 \longrightarrow \boldsymbol{x}_T$



Diffusion models as SDEs

x_0 $\xrightarrow{\mathrm{d}\boldsymbol{X}_t = \boldsymbol{f}(\boldsymbol{X}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{W}_t}$ x_T



x_0 $\xleftarrow{\quad}$ x_T

$$\mathrm{d}\boldsymbol{X}_\tau = [-\boldsymbol{f}(\boldsymbol{X}_\tau, \tau) + g^2(\tau)\nabla_{\boldsymbol{x}} \log p_\tau(\boldsymbol{X}_\tau)]\,\mathrm{d}\tau + g(\tau)\,\mathrm{d}\overline{\boldsymbol{W}}_\tau$$

Proposition 7 (Cœfficients of Gaussian processes with fixed perturbation kernel). Given a linear Itô SDE,

$$\mathrm{d}\boldsymbol{X}_t = f(t)\,\mathrm{d}t\,\boldsymbol{X}_t + g(t)\,\mathrm{d}\boldsymbol{W}_t, \tag{32}$$

a strictly monotonically decreasing function $\alpha_t \in C^\infty(\mathbb{R};\mathbb{R}_{\geq 0})$, a strictly monotonically increasing function $\sigma_t \in C^\infty(\mathbb{R};\mathbb{R}_{\geq 0})$, with boundary conditions $\alpha_0 = 1$ and $\sigma_0 = 0$; and desired transition kernel for the Itô process described by the SDE of the form

$$q_{t|0}(\boldsymbol{x}_t|\boldsymbol{x}_0) = p_{\mathcal{N}}(\boldsymbol{x}_t; \alpha_t \boldsymbol{x}_0, \sigma_t^2 \boldsymbol{I}), \tag{33}$$

then the drift and diffusion cœfficients for the SDE are

$$f(t) = \frac{\mathrm{d}\log \alpha_t}{\mathrm{d}t}, \quad g^2(t) = \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} - 2\sigma_t^2 \frac{\mathrm{d}\log \alpha_t}{\mathrm{d}t}. \tag{34}$$

Reversible solvers for SDEs

- The only reversible solver for SDEs is Kidger's reversible Heun

- The only reversible solver for SDEs is Kidger's reversible Heun
- We extend to the McCallum-Foster method to SDEs

- The only reversible solver for SDEs is Kidger's reversible Heun
- We extend to the McCallum-Foster method to SDEs
- And we develop a bespoke reversible solver for diffusion models

Overview of reversible solvers for diffusion SDEs

Scheme	Bespoke solver	Memory	ODE Stability
Reversible Heun (NeurIPS 2021)[16]	✗	$\mathcal{O}(T)$	✗
CycleDiffusion (ICCV 2023)[17]	✓	$\mathcal{O}(dT)$	✗
Ours (ICLR 2025)[18]	✓	$\mathcal{O}(T)$	✓

16 P. Kidger et al. (2021). "Efficient and accurate gradients for neural sdes". In: *Advances in Neural Information Processing Systems* 34, pp. 18747–18761.

17 C. H. Wu and F. D. la Torre (2023). "A Latent Space of Stochastic Diffusion Models for Zero-Shot Image Editing and Guidance". In: *ICCV*.

18 Z. W. Blasingame and C. Liu (2025a). "A Reversible Solver for Diffusion SDEs". In: *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*. URL: `https://openreview.net/forum?id=0gEFLVUL6n`.

What to do with the noise?

- Integrating in forward-time and reverse-time with SDEs is hard

[19] T. J. Lyons (1998). "Differential equations driven by rough signals". In: *Revista Matemática Iberoamericana* 14.2, pp. 215–310.

What to do with the noise?

- Integrating in forward-time and reverse-time with SDEs is hard
- A pathwise interpretation of SDEs can make this easier

[19] T. J. Lyons (1998). "Differential equations driven by rough signals". In: *Revista Matemática Iberoamericana* 14.2, pp. 215–310.

What to do with the noise?

- Integrating in forward-time and reverse-time with SDEs is hard
- A pathwise interpretation of SDEs can make this easier
- We make use of Terry Lyons' theory of rough paths[19]

[19] T. J. Lyons (1998). "Differential equations driven by rough signals". In: *Revista Matemática Iberoamericana* 14.2, pp. 215–310.

What to do with the noise?

- Integrating in forward-time and reverse-time with SDEs is hard
- A pathwise interpretation of SDEs can make this easier
- We make use of Terry Lyons' theory of rough paths[19]
- This allows us to discuss a stochastic process X_t pathwise

[19] T. J. Lyons (1998). "Differential equations driven by rough signals". In: *Revista Matemática Iberoamericana* 14.2, pp. 215–310.

What to do with the noise?

- Integrating in forward-time and reverse-time with SDEs is hard
- A pathwise interpretation of SDEs can make this easier
- We make use of Terry Lyons' theory of rough paths[19]
- This allows us to discuss a stochastic process $\boldsymbol{X}_t$ pathwise
- Fix an ω, then there is a deterministic map $\boldsymbol{W}_t(\omega) \mapsto \boldsymbol{X}_t(\omega)$

[19] T. J. Lyons (1998). "Differential equations driven by rough signals". In: *Revista Matemática Iberoamericana* 14.2, pp. 215–310.

What to do with the noise?

- Integrating in forward-time and reverse-time with SDEs is hard
- A pathwise interpretation of SDEs can make this easier
- We make use of Terry Lyons' theory of rough paths[19]
- This allows us to discuss a stochastic process $\boldsymbol{X}_t$ pathwise
- Fix an ω, then there is a deterministic map $\boldsymbol{W}_t(\omega) \mapsto \boldsymbol{X}_t(\omega)$
- How do we find $\boldsymbol{W}_t$ in reverse-time?

[19] T. J. Lyons (1998). "Differential equations driven by rough signals". In: *Revista Matemática Iberoamericana* 14.2, pp. 215–310.

What to do with the noise?

- Integrating in forward-time and reverse-time with SDEs is hard
- A pathwise interpretation of SDEs can make this easier
- We make use of Terry Lyons' theory of rough paths[19]
- This allows us to discuss a stochastic process $\boldsymbol{X}_t$ pathwise
- Fix an ω, then there is a deterministic map $\boldsymbol{W}_t(\omega) \mapsto \boldsymbol{X}_t(\omega)$
- How do we find $\boldsymbol{W}_t$ in reverse-time?
- Naïve solution is to just store $\boldsymbol{W}_t$ entirely in memory

[19] T. J. Lyons (1998). "Differential equations driven by rough signals". In: *Revista Matemática Iberoamericana* 14.2, pp. 215–310.

What to do with the noise?

- Integrating in forward-time and reverse-time with SDEs is hard
- A pathwise interpretation of SDEs can make this easier
- We make use of Terry Lyons' theory of rough paths[19]
- This allows us to discuss a stochastic process $\boldsymbol{X}_t$ pathwise
- Fix an ω, then there is a deterministic map $\boldsymbol{W}_t(\omega) \mapsto \boldsymbol{X}_t(\omega)$
- How do we find $\boldsymbol{W}_t$ in reverse-time?
- Naïve solution is to just store $\boldsymbol{W}_t$ entirely in memory
- Let's not do that

[19] T. J. Lyons (1998). "Differential equations driven by rough signals". In: *Revista Matemática Iberoamericana* 14.2, pp. 215–310.

The Brownian Interval

- Let $\boldsymbol{W}_{s,t} = \boldsymbol{W}_t - \boldsymbol{W}_s$ denote the Brownian interval on $[s, t]$

[20] P. Kidger (2022). "On Neural Differential Equations". Available at `https://arxiv.org/abs/2202.02435`. Ph.D. thesis. Oxford University.

[21] K. Clæssen and M. H. Pałka (2013). "Splittable pseudorandom number generators using cryptographic hashing". In: *ACM SIGPLAN Notices* 48.12, pp. 47–58.

The Brownian Interval

- Let $\boldsymbol{W}_{s,t} = \boldsymbol{W}_t - \boldsymbol{W}_s$ denote the Brownian interval on $[s, t]$
- Kidger et al.[20] propose an algorithm using a splittable PRNG[21]

[20] P. Kidger (2022). “On Neural Differential Equations”. Available at `https://arxiv.org/abs/2202.02435`. Ph.D. thesis. Oxford University.

[21] K. Clæssen and M. H. Pałka (2013). “Splittable pseudorandom number generators using cryptographic hashing”. In: *ACM SIGPLAN Notices* 48.12, pp. 47–58.

The Brownian Interval

- Let $\boldsymbol{W}_{s,t} = \boldsymbol{W}_t - \boldsymbol{W}_s$ denote the Brownian interval on $[s, t]$
- Kidger et al.[20] propose an algorithm using a splittable PRNG[21]
- We want to find $\boldsymbol{W}_{s,t}$, $0 < s < t < T$

[20] P. Kidger (2022). "On Neural Differential Equations". Available at https://arxiv.org/abs/2202.02435. Ph.D. thesis. Oxford University.

[21] K. Claessen and M. H. Pałka (2013). "Splittable pseudorandom number generators using cryptographic hashing". In: *ACM SIGPLAN Notices* 48.12, pp. 47–58.

The Brownian Interval

$$[0, T]$$

- Let $\boldsymbol{W}_{s,t} = \boldsymbol{W}_t - \boldsymbol{W}_s$ denote the Brownian interval on $[s, t]$
- Kidger et al.[20] propose an algorithm using a splittable PRNG[21]
- We want to find $\boldsymbol{W}_{s,t}$, $0 < s < t < T$
- Start with $\boldsymbol{W}_{0,T} = \boldsymbol{W}_T$

[20] P. Kidger (2022). "On Neural Differential Equations". Available at https://arxiv.org/abs/2202.02435. Ph.D. thesis. Oxford University.

[21] K. Clæssen and M. H. Pałka (2013). "Splittable pseudorandom number generators using cryptographic hashing". In: *ACM SIGPLAN Notices* 48.12, pp. 47–58.

The Brownian Interval

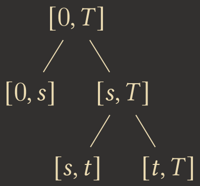
$$\begin{tikzcd}[column sep=small] & {[0, T]} \arrow[dl, dash] \arrow[dr, dash] & \\ {[0, s]} & & {[s, T]} \end{tikzcd}$$

- Let $\boldsymbol{W}_{s,t} = \boldsymbol{W}_t - \boldsymbol{W}_s$ denote the Brownian interval on $[s, t]$
- Kidger et al.[20] propose an algorithm using a splittable PRNG[21]
- We want to find $\boldsymbol{W}_{s,t}$, $0 < s < t < T$
- Start with $\boldsymbol{W}_{0,T} = \boldsymbol{W}_T$
- Use Lévy's Brownian bridge formula to find $\boldsymbol{W}_{0,s}$ and $\boldsymbol{W}_{s,T}$

[20] P. Kidger (2022). "On Neural Differential Equations". Available at `https://arxiv.org/abs/2202.02435`. Ph.D. thesis. Oxford University.

[21] K. Clæssen and M. H. Pałka (2013). "Splittable pseudorandom number generators using cryptographic hashing". In: *ACM SIGPLAN Notices* 48.12, pp. 47–58.

The Brownian Interval



- Let $W_{s,t} = W_t - W_s$ denote the Brownian interval on $[s, t]$
- Kidger et al.[20] propose an algorithm using a splittable PRNG[21]
- We want to find $W_{s,t}$, $0 < s < t < T$
- Start with $W_{0,T} = W_T$
- Use Lévy's Brownian bridge formula to find $W_{0,s}$ and $W_{s,T}$
- Repeat to find $W_{s,t}$

---

[20] P. Kidger (2022). "On Neural Differential Equations". Available at `https://arxiv.org/abs/2202.02435`. Ph.D. thesis. Oxford University.

[21] K. Clæssen and M. H. Pałka (2013). "Splittable pseudorandom number generators using cryptographic hashing". In: *ACM SIGPLAN Notices* 48.12, pp. 47–58.

Towards a bespoke reversible SDE solver with target prediction

- Recall the reverse-time diffusion SDE.

$$\mathrm{d}\boldsymbol{X}_t = \left[f(t)\boldsymbol{X}_t - g^2(t)\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{X}_t)\right] \mathrm{d}t + g(t) \circ \mathrm{d}\overline{\boldsymbol{W}}_t \tag{35}$$

Towards a bespoke reversible SDE solver with target prediction

- Recall the reverse-time diffusion SDE.
- Recall the definition of target prediction

$$\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) = -\frac{1}{\sigma_t^2}\boldsymbol{x} + \frac{\alpha_t}{\sigma_t^2}\boldsymbol{x}_{0|t}(\boldsymbol{x}).$$

$$\mathrm{d}\boldsymbol{X}_t = [f(t)\boldsymbol{X}_t - g^2(t)\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{X}_t)]\,\mathrm{d}t + g(t) \circ \mathrm{d}\overline{\boldsymbol{W}}_t \tag{35}$$

Towards a bespoke reversible SDE solver with target prediction

- Recall the reverse-time diffusion SDE.
- Recall the definition of target prediction
- Then with a little algebra,

$$\mathrm{d}\boldsymbol{X}_t = \left[\left(f(t) + \frac{g^2(t)}{\sigma_t^2}\right)\boldsymbol{X}_t - \frac{\alpha_t}{\sigma_t^2} g^2(t) \boldsymbol{x}_{0|t}(\boldsymbol{X}_t)\right] \mathrm{d}t + g(t) \circ \mathrm{d}\overline{\boldsymbol{W}}_t \tag{35}$$

Proposition 8 (Exact solution of diffusion SDEs with target prediction). Given an initial value $\boldsymbol{X}_s = \boldsymbol{x}_s$ at time $s \in [0, T]$ the exact solution of Eq. (35) can be expressed as

$$\boldsymbol{X}_t = \underbrace{\frac{\sigma_t}{\sigma_s} e^{\lambda_s - \lambda_t} \boldsymbol{X}_s}_{\substack{\textbf{Linear term} \\ \text{No truncation errors}}} + \underbrace{2\alpha_t \int_{\lambda_s}^{\lambda_t} e^{2(\lambda - \lambda_t)} \boldsymbol{x}_{0|\lambda}(\boldsymbol{X}_\lambda)\, \mathrm{d}\lambda}_{\substack{\textbf{Approximated term} \\ \text{Truncation errors}}} + \underbrace{\sqrt{2}\sigma_t e^{-\lambda_t} \boldsymbol{W}_{\varsigma_s, \varsigma_t}}_{\substack{\textbf{Brownian interval} \\ \text{No truncation errors}}}, \tag{36}$$

where $\varsigma_t = \frac{1}{2}(e^{2\lambda_t} - e^{2\lambda_T})$ and $\{\boldsymbol{W}_t : 0 \le t \le T\}$ is the standard Brownian motion.

A reversible solver for diffusion SDEs

- Let Ψ_h denote an explicit one-step method for the integral term in Eq. (36)
- Let $\zeta \in (0, 1]$ denote a coupling parameter used for stability
- Let $h := \lambda_t - \lambda_s$ be the step size in the log-SNR domain
- Initialize $\hat{\boldsymbol{x}}_0 = \boldsymbol{x}_0$

Forward step

$$\boldsymbol{x}_{n+1} = \zeta \boldsymbol{x}_n + (1 - \zeta)\hat{\boldsymbol{x}}_n + \frac{\sigma_{n+1}}{\sigma_n} e^{-h} \hat{\boldsymbol{x}}_n + 2\alpha_{n+1} \Psi_h(t_n, \hat{\boldsymbol{x}}_n) + \sqrt{2}\sigma_{n+1} e^{-\lambda_{n+1}} \boldsymbol{W}_{\varsigma_n, \varsigma_{n+1}}$$

$$\hat{\boldsymbol{x}}_{n+1} = \hat{\boldsymbol{x}}_n - \frac{\sigma_n}{\sigma_{n+1}} e^{h} \boldsymbol{x}_{n+1} - 2\alpha_n \Psi_{-h}(t_{n+1}, \boldsymbol{x}_{n+1}) + \sqrt{2}\sigma_n e^{-\lambda_n} \boldsymbol{W}_{\varsigma_n, \varsigma_{n+1}}$$

A reversible solver for diffusion SDEs

- Let Ψ_h denote an explicit one-step method for the integral term in Eq. (36)
- Let $\zeta \in (0, 1]$ denote a coupling parameter used for stability
- Let $h := \lambda_t - \lambda_s$ be the step size in the log-SNR domain
- Initialize $\hat{\boldsymbol{x}}_0 = \boldsymbol{x}_0$

Backward step

$$
\begin{aligned}
\hat{\boldsymbol{x}}_n &= \hat{\boldsymbol{x}}_{n+1} + \frac{\sigma_n}{\sigma_{n+1}} e^h \boldsymbol{x}_{n+1} + 2\alpha_n \Psi_{-h}(t_{n+1}, \boldsymbol{x}_{n+1}) - \sqrt{2}\sigma_n e^{-\lambda_n} \boldsymbol{W}_{\varsigma_n, \varsigma_{n+1}} \\
\boldsymbol{x}_n &= \zeta^{-1} \boldsymbol{x}_{n+1} + (1 - \zeta^{-1}) \hat{\boldsymbol{x}}_n - \frac{\sigma_{n+1}}{\sigma_n} e^{-h} \zeta^{-1} \hat{\boldsymbol{x}}_n + 2\alpha_{n+1} \zeta^{-1} \Psi_h(t_n, \hat{\boldsymbol{x}}_n) \\
&\quad - \sqrt{2}\sigma_{n+1} e^{-\lambda_{n+1}} \zeta^{-1} \boldsymbol{W}_{\varsigma_n, \varsigma_{n+1}}
\end{aligned}
$$

An illustration

(a) DDIM inversion with 20 steps

(b) Reversible DDIM with 20 steps

(c) Reversible diffusion SDE with 20 steps

Conclusion

Summary

End-to-end guidance

- Developed a family of bespoke numerical solvers for the continuous adjoint equations of flow/diffusion models

Summary

End-to-end guidance

- Developed a family of bespoke numerical solvers for the continuous adjoint equations of flow/diffusion models
- Showed how to find the gradients of a time-dependent conditioning signal

Summary

End-to-end guidance

- Developed a family of bespoke numerical solvers for the continuous adjoint equations of flow/diffusion models
- Showed how to find the gradients of a time-dependent conditioning signal
- We showed that the continuous adjoint equations for additive noise SDEs are actually ODEs

Summary

End-to-end guidance

- Developed a family of bespoke numerical solvers for the continuous adjoint equations of flow/diffusion models
- Showed how to find the gradients of a time-dependent conditioning signal
- We showed that the continuous adjoint equations for additive noise SDEs are actually ODEs

Greedy guidance

- We connect posterior and end-to-end guidance in a unified framework

Summary

End-to-end guidance

- Developed a family of bespoke numerical solvers for the continuous adjoint equations of flow/diffusion models
- Showed how to find the gradients of a time-dependent conditioning signal
- We showed that the continuous adjoint equations for additive noise SDEs are actually ODEs

Greedy guidance

- We connect posterior and end-to-end guidance in a unified framework
- We show that a greedy strategy makes good decisions

Summary

End-to-end guidance

- Developed a family of bespoke numerical solvers for the continuous adjoint equations of flow/diffusion models
- Showed how to find the gradients of a time-dependent conditioning signal
- We showed that the continuous adjoint equations for additive noise SDEs are actually ODEs

Greedy guidance

- We connect posterior and end-to-end guidance in a unified framework
- We show that a greedy strategy makes good decisions
- Provide some justifications for why greed is good

Summary

End-to-end guidance

- Developed a family of bespoke numerical solvers for the continuous adjoint equations of flow/diffusion models
- Showed how to find the gradients of a time-dependent conditioning signal
- We showed that the continuous adjoint equations for additive noise SDEs are actually ODEs

Greedy guidance

- We connect posterior and end-to-end guidance in a unified framework
- We show that a greedy strategy makes good decisions
- Provide some justifications for why greed is good

Reversible solvers

- We developed a reversible solver for SDEs with exact inversion and low distortion

Summary

End-to-end guidance

- Developed a family of bespoke numerical solvers for the continuous adjoint equations of flow/diffusion models
- Showed how to find the gradients of a time-dependent conditioning signal
- We showed that the continuous adjoint equations for additive noise SDEs are actually ODEs

Greedy guidance

- We connect posterior and end-to-end guidance in a unified framework
- We show that a greedy strategy makes good decisions
- Provide some justifications for why greed is good

Reversible solvers

- We developed a reversible solver for SDEs with exact inversion and low distortion
- We show that prior diffusion methods for exact inversion are just the midpoint method

Future directions

- Compute vs accuracy of gradients.

Future directions

- Compute vs accuracy of gradients.
- Parallelized strategies for gradient computation.

Future directions

- Compute vs accuracy of gradients.
- Parallelized strategies for gradient computation.
- Gradient-free guidance.

Future directions

- Compute vs accuracy of gradients.
- Parallelized strategies for gradient computation.
- Gradient-free guidance.
- Reversible solvers for continuous-time discrete-space processes.

Future directions

- Compute vs accuracy of gradients.
- Parallelized strategies for gradient computation.
- Gradient-free guidance.
- Reversible solvers for continuous-time discrete-space processes.
- Applications of reversible solvers to latent editing in AI4Science applications.

?

References

Pontryagin, L. S., Boltyanskii, V. G., Gamkrelidze, R. V., and Mishechenko, E. F. (1963). "The Mathematical Theory of Optimal Processes.". In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 43.10-11, pp. 514–515. DOI: `https://doi.org/10.1002/zamm.19630431023`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/zamm.19630431023`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/zamm.19630431023` (cit. on pp. 47–49).

Lyons, T. J. (1998). "Differential equations driven by rough signals". In: *Revista Matemática Iberoamericana* 14.2, pp. 215–310 (cit. on pp. 114–121).

Clæssen, K. and Pałka, M. H. (2013). "Splittable pseudorandom number generators using cryptographic hashing". In: *ACM SIGPLAN Notices* 48.12, pp. 47–58 (cit. on pp. 122–127).

DeBruine, L. and Jones, B. (May 2017). *Face Research Lab London Set*. DOI: 10.6084/m9.figshare.5047666.v5. URL: https://figshare.com/articles/dataset/Face_Research_Lab_London_Set/5047666 (cit. on pp. 89, 90).

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). “Neural ordinary differential equations”. In: *Advances in neural information processing systems* 31 (cit. on pp. 47–49).

Torres, G., Yang, Z., Blasingame, Z., Bruska, J., and Liu, C. (June 2019). “Detecting Non-Control-Flow Hijacking Attacks Using Contextual Execution Information”. In: *Proceedings of the 8th International Workshop on Hardware and Architectural Support for Security and Privacy*. HASP ’19. Phœnix, AZ, USA: Association for Computing Machinery. ISBN: 9781450372268. DOI: 10.1145/3337167.3337168. URL: https://doi.org/10.1145/3337167.3337168 (cit. on p. 6).

- Blasingame, Z. and Liu, C. (2021). “Leveraging adversarial learning for the detection of morphing attacks”. In: *2021 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE, pp. 1–8 (cit. on p. 4).
- Blasingame, Z., Liu, C., and Yao, X. (2021). “Feature Creation Towards the Detection of Non-control-Flow Hijacking Attacks”. In: *International Conference on Artificial Neural Networks*. Springer, pp. 153–164 (cit. on p. 4).
- Kidger, P., Foster, J., Li, X. C., and Lyons, T. (2021). “Efficient and accurate gradients for neural sdes”. In: *Advances in Neural Information Processing Systems* 34, pp. 18747–18761 (cit. on pp. 104, 113).
- Zhang, H., Venkatesh, S., Ramachandra, R., Raja, K., Damer, N., and Busch, C. (2021). “MIPGAN—Generating Strong and High Quality Morphing Attacks Using Identity Prior Driven GAN”. In: *IEEE Transactions on Biometrics, Behavior, and Identity Science* 3.3, pp. 365–383. DOI: `10.1109/TBIOM.2021.3072349` (cit. on p. 91).

- Zhuang, J., Dvornek, N. C., tatikonda, sekhar, and Duncan, J. s (2021). “MALI: A memory efficient and reverse accurate integrator for Neural ODEs”. In: *International Conference on Learning Representations*. URL: `https://openreview.net/forum?id=blfSjHeFM_e` (cit. on p. 104).

- Huber, M., Boutros, F., Luu, A. T., Raja, K., Ramachandra, R., Damer, N., Neto, P. C., Gonçalves, T., Sequeira, A. F., Cardoso, J. S., Tremoço, J., Lourenço, M., Serra, S., Cermeño, E., Ivanovska, M., Batagelj, B., Kronovšek, A., Peer, P., and Štruc, V. (2022). “SYN-MAD 2022: Competition on Face Morphing Attack Detection Based on Privacy-aware Synthetic Training Data”. In: *2022 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 1–10. DOI: `10.1109/IJCB54206.2022.10007950` (cit. on p. 91).

- Kidger, P. (2022). “On Neural Differential Equations”. Available at `https://arxiv.org/abs/2202.02435`. Ph.D. thesis. Oxford University (cit. on pp. 122–127).

Nie, W., Guo, B., Huang, Y., Xiao, C., Vahdat, A., and Anandkumar, A. (17–23 Jul 2022). "Diffusion Models for Adversarial Purification". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 16805–16827. URL: https://proceedings.mlr.press/v162/nie22a.html (cit. on p. 46).

Woralert, C., Liu, C., and Blasingame, Z. (2022). "HARD-Lite: A Lightweight Hardware Anomaly Realtime Detection Framework Targeting Ransomware". In: *2022 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, pp. 1–6 (cit. on p. 5).

Bansal, A., Chu, H.-M., Schwarzschild, A., Sengupta, S., Goldblum, M., Geiping, J., and Goldstein, T. (2023). "Universal guidance for diffusion models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 843–852 (cit. on p. 75).

References vi

Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. (2023). "Diffusion Posterior Sampling for General Noisy Inverse Problems". In: *The Eleventh International Conference on Learning Representations, ICLR 2023*. The International Conference on Learning Representations (cit. on p. 75).

Liu, X., Wu, L., Zhang, S., Gong, C., Ping, W., and Liu, Q. (2023). "Flowgrad: Controlling the output of generative odes with gradients". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24335–24344 (cit. on p. 75).

Wallace, B., Gokul, A., and Naik, N. (2023). "Edict: Exact diffusion inversion via coupled transformations". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22532–22541 (cit. on p. 104).

Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., et al. (2023). "De novo design of protein structure and function with RFdiffusion". In: *Nature* 620.7976, pp. 1089–1100 (cit. on pp. 11–13).

- Woralert, C., Liu, C., and Blasingame, Z. (2023). “Hard-lite: A lightweight hardware anomaly realtime detection framework targeting ransomware”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* (cit. on p. 5).
- Woralert, C., Liu, C., Blasingame, Z., and Yang, Z. (2023). “A Comparison of One-class and Two-class Models for Ransomware Detection via Low-level Hardware Information”. In: *2023 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, pp. 1–6 (cit. on p. 5).
- Wu, C. H. and Torre, F. D. la (2023). “A Latent Space of Stochastic Diffusion Models for Zero-Shot Image Editing and Guidance”. In: *ICCV* (cit. on p. 113).
- Yu, J., Wang, Y., Zhao, C., Ghanem, B., and Zhang, J. (2023). “Freedom: Training-free energy-guided conditional diffusion model”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23174–23184 (cit. on p. 75).
- Zhang, H., Ramachandra, R., Raja, K., and Christoph, B. (2023). “Morph-PIPE: Plugging in Identity Prior to Enhance Face Morphing Attack Based on Diffusion Model”. In: *Norwegian Information Security Conference (NISK)* (cit. on p. 91).

- Ben-Hamu, H., Puny, O., Gat, I., Karrer, B., Singer, U., and Lipman, Y. (2024). "D-Flow: Differentiating through Flows for Controlled Generation". In: *Forty-first International Conference on Machine Learning*. URL: https://openreview.net/forum?id=SE2OBFqj6J (cit. on pp. 46, 75, 92, 93).
- , B. (2024). *FLUX*. https://github.com/black-forest-labs/flux (cit. on pp. 9–13).
- Blasingame, Z. W. and Liu, C. (2024a). "AdjointDEIS: Efficient Gradients for Diffusion Models". In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang. Vol. 37. Curran Associates, Inc., pp. 2449–2483. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/04badd3b048315c8c3a0ca17eff723d7-Paper-Conference.pdf (cit. on pp. 3, 40, 46, 75, 91).
- — (2024b). "Fast-dim: Towards fast diffusion morphs". In: *IEEE Security & Privacy* (cit. on pp. 4, 91).

Blasingame, Z. W. and Liu, C. (2024c). “Greedy-DiM: Greedy Algorithms for Unreasonably Effective Face Morphs”. In: *2024 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 1–11. DOI: 10.1109/IJCB62174.2024.10744517 (cit. on pp. 3, 91).

— (2024d). “Leveraging diffusion for strong and high quality face morphing attacks”. In: *IEEE Transactions on Biometrics, Behavior, and Identity Science* 6.1, pp. 118–131 (cit. on pp. 4, 88, 91).

McCallum, S. and Foster, J. (2024). “Efficient, Accurate and Stable Gradients for Neural ODEs”. In: *arXiv preprint arXiv:2410.11648* (cit. on p. 104).

Neddo, R. E., Blasingame, Z. W., and Liu, C. (2024). “The Impact of Print-Scanning in Heterogeneous Morph Evaluation Scenarios”. In: *2024 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 1–10. DOI: 10.1109/IJCB62174.2024.10744441 (cit. on p. 5).

Nie, S., Guo, H. A., Lu, C., Zhou, Y., Zheng, C., and Li, C. (2024). "The Blessing of Randomness: SDE Beats ODE in General Diffusion-based Image Editing". In: *The Twelfth International Conference on Learning Representations*. URL: https://openreview.net/forum?id=DesYwmUG00 (cit. on p. 105).

Pan, J., Liew, J. H., Tan, V., Feng, J., and Yan, H. (2024). "AdjointDPM: Adjoint Sensitivity Method for Gradient Backpropagation of Diffusion Probabilistic Models". In: *The Twelfth International Conference on Learning Representations*. URL: https://openreview.net/forum?id=y331DRBgWI (cit. on pp. 46, 75).

Schneider, F., Kamal, O., Jin, Z., and Schölkopf, B. (2024). "Moûsai: Efficient text-to-music diffusion models". In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8050–8068 (cit. on pp. 10–13).

- Wang, F., Yin, H., Dong, Y.-J., Zhu, H., Zhang, C., Zhao, H., Qian, H., and Li, C. (2024). "BELM: Bidirectional Explicit Linear Multi-step Sampler for Exact Inversion in Diffusion Models". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. URL: https://openreview.net/forum?id=ccQ4fmwLDb (cit. on p. 104).
- Woralert, C., Liu, C., and Blasingame, Z. (2024). "Towards Effective Machine Learning Models for Ransomware Detection via Low-Level Hardware Information". In: *Proceedings of the International Workshop on Hardware and Architectural Support for Security and Privacy 2024*, pp. 10–18 (cit. on p. 5).
- Zhang, G., Lewis, J. P., and Kleijn, W. B. (2024). "Exact Diffusion Inversion via Bidirectional Integration Approximation". In: *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LVII*. Milan, Italy: Springer-Verlag, pp. 19–36. ISBN: 978-3-031-72997-3. DOI: 10.1007/978-3-031-72998-0_2. URL: https://doi.org/10.1007/978-3-031-72998-0_2 (cit. on p. 104).

- Blasingame, Z. W. and Liu, C. (2025a). "A Reversible Solver for Diffusion SDEs". In: *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*. URL: https://openreview.net/forum?id=0gEFLVUL6n (cit. on pp. 3, 96, 113).
- — (2025b). "Greed is Good: Guided Generation from a Greedy Perspective". In: *Frontiers in Probabilistic Inference: Learning meets Sampling*. URL: https://openreview.net/forum?id=o4yQzZ5qCW (cit. on pp. 3, 71).
- Marion, P., Korba, A., Bartlett, P., Blondel, M., Bortoli, V. D., Doucet, A., Llinares-López, F., Paquette, C., and Berthet, Q. (2025). "Implicit Diffusion: Efficient optimization through stochastic sampling". In: *The 28th International Conference on Artificial Intelligence and Statistics*. URL: https://openreview.net/forum?id=r5F7Z8s0Qk (cit. on pp. 46, 75).
- Neddo, R. E., Willis, S., Blasingame, Z. W., and Liu, C. (2025). "Poster: Adapting Pretrained Vision Transformers with LoRA Against Attack Vectors". In: *2025 IEEE International Conference on Mobility, Operations, Services, and Technologies (MOST)* (cit. on p. 4).

Wang, L., Cheng, C., Liao, Y., Qu, Y., and Liu, G. (2025). “Training Free Guided Flow-Matching with Optimal Control”. In: *The Thirteenth International Conference on Learning Representations*. URL: https://openreview.net/forum?id=61ss5RA1MM (cit. on pp. 46, 75).