Proceedings of the 7th Symposium on Advances in Approximate Bayesian Inference, 2025 1–22

Neural Flow Samplers with Shortcut Models

Wuhao Chen* wuhao.chen21@imperial.ac.uk
Zijing Ou* z.ou22@imperial.ac.uk
Yingzhen Li yingzhen.li@imperial.ac.uk
Imperial College London

Abstract

Sampling from unnormalized densities is a fundamental task across various domains. Flow-based samplers generate samples by learning a velocity field that satisfies the continuity equation, but this requires estimating the intractable time derivative of the partition function. While importance sampling provides an approximation, it suffers from high variance. To mitigate this, we introduce a velocity-driven Sequential Monte Carlo method combined with control variates to reduce variance. Additionally, we incorporate a shortcut model to improve efficiency by minimizing the number of sampling steps. Empirical results on both synthetic datasets and n-body system targets validate the effectiveness of our approach.

1. Introduction

We consider the task of sampling from unnormalised densities $\pi = \frac{\rho}{Z}$, where $Z := \int \rho(x)\,\mathrm{d}x$ denotes the partition function. This task is fundamental in probabilistic modelling and scientific simulations, with broad applications in Bayesian inference (Neal, 1993), nuclear physics (Albergo et al., 2019), drug discovery (Xie et al., 2021), and material design (Komanduri et al., 2000). However, achieving efficient sampling remains challenging, especially when dealing with high-dimensional and multi-modal distributions.

Conventional sampling methods rely on Markov Chain Monte Carlo (MCMC) (Neal, 2012), which requires long convergence times and the simulation of extended chains to obtain uncorrelated samples. Neural samplers address these issues by approximating the target distribution using generative models, such as normalizing flows (Midgley et al., 2023) and latent variable models (He et al., 2024). Building on the success of diffusion models, a diffusion-based sampler (Sadegh et al., 2024) has been introduced to train a score network that approximates the estimated score through Monte Carlo estimation. Concurrently, Tian et al. (2024) propose a flow-based sampler that learns a velocity model to satisfy the continuity equation (Villani et al., 2009). These approaches show strong empirical performance, offering more efficient and scalable alternatives to MCMC samplers.

In this work, we focus on training flow-based samplers, which present challenges due to the intractable time derivative of the logarithm of the partition function in the continuity equation. While Tian et al. (2024) use importance sampling to approximate it, the approach suffers from high variance, limiting its effectiveness. To address this, we introduce a more stable and efficient estimation method, improving both the accuracy and efficiency of flow-based samplers. Specifically, we propose a velocity-driven sequential Monte Carlo (VD-SMC) (Del Moral et al., 2006) combined with control variates (Geffner and Domke, 2018) to reduce

* Equal contribution.

variance. VD-SMC operates in a bootstrap manner, generating high-quality training samples while producing low-variance estimates of the time derivative. To further enhance sampling efficiency, we incorporate the shortcut model (Frans et al., 2024), a self-distillation technique that reduces the number of required sampling steps. Empirical evaluations on both synthetic and n-body system targets demonstrate the effectiveness of our approach.

2. Background: Continuity Equation

In this section, we introduce the key preliminary: continuity equation. Let $p_t, t \in [0,1]$ be the probability path on $\mathbb{R}^d$, we say the velocity $v_t : \mathbb{R}^d \to \mathbb{R}^d$ generates the path p_t if the continuity equation holds: $\partial_t p_t(x) = -\nabla_x \cdot [p_t(x) v_t(x)], \forall x \in \mathbb{R}^d$, where $\nabla_x \cdot$ denotes the divergence operator. Thus sampling from the path p_t can be done by solving the integral $x_t = x_0 + \int_{s=0}^{t} v_s(x_s)\,\mathrm{d}s$, with $x_0 \sim p_0$. Dividing both sides of the continuity equation by p_t leads to

$$\partial_t \log p_t(x) = -\nabla_x \cdot v_t(x) - v_t(x) \cdot \nabla_x \log p_t(x), \forall x \in \mathbb{R}^d. \tag{1}$$

This equation further leads to the instantaneous change of variable

$$\partial_t [\log p_t(x_t)] = \partial_t \log p_t(x_t) + \nabla_{x_t} \log p_t(x_t) \cdot v_t(x_t) = -\nabla_{x_t} \cdot v_t(x_t), \tag{2}$$

where $\partial_t [\log p_t(x_t)]$ denotes the total derivative w.r.t. t and we apply the fact that $\partial_t x_t = v_t(x_t)$. Thus, Equation (2) can be used to evaluate the log-likelihood of the sample $x_1 \sim p_1$. Next, we introduce how to employ Equation (1) to learn a model-based velocity for sampling from the target density π, followed by the further improvement of using shortcut models.

3. Neural Flow Sampler

To learn a model-based velocity $v_t(x;\theta)$, parametrised by θ, we first define the probability path p_t. Specifically, given a tractable base distribution η, the path is constructed using annealing interpolation as $p_t \propto \rho^t \eta^{1-t} =: \tilde{p}_t$. The velocity can then be learned by minimizing the following loss

$$\mathcal{L}(\theta) = \mathbb{E}_{q(x),w(t)} \delta_t^2(x; v_t(\cdot;\theta)), \quad \delta_t(x; v_t) \triangleq \partial_t \log p_t(x) + \nabla_x \cdot v_t(x) + v_t(x) \cdot \nabla_x \log p_t(x), \tag{3}$$

where $q(x)$ is any distribution has the same support of the target π and $w(t)$ denotes the time schedule distribution. This objective, initially proposed by Tian et al. (2024), poses several challenges. First, computing the divergence $\nabla_x \cdot v_t(x)$ can be prohibitive in high-dimension; however, this issue can be mitigated using the Hutchinson estimator (Hutchinson, 1989). More critically, the time derivative introduces additional complexity, as it involves $\partial_t \log p_t(x) = \partial_t \log \tilde{p}_t(x) - \partial_t \log Z_t$ with $Z_t = \int \tilde{p}_t(x)\,\mathrm{d}x$, which is intractable. Tian et al. (2024) estimate it using importance sampling $\partial_t \log Z_t \approx \sum_k \frac{w_k}{\sum_k w_k} \partial_t \log \tilde{p}_t(x_t^{(k)})$, where $x_t^{(k)} \sim p_t(x;\theta)$ denotes the sample generated by the velocity $v_t(x;\theta)$ at time t, and $\log w_t = \int_0^t \delta_s(x_s; v_s(\cdot;\theta))\,\mathrm{d}s$ (see Appendix A.1 for details). However, importance sampling can suffer from high variance if the proposal $p_t(x;\theta)$ differs significantly from the target $p_t(x)$, leading to a low effective sample size. In the following, we propose a velocity-driven method to estimate the intractable time derivatives $\partial_t \log Z_t$.

3.1. Velocity-Driven Sequential Monte Carlo Estimation

As discussed before, even though importance sampling provides a simple way to approximate the intractable time derivatives, it often suffers from high variance. To alleviate this issue, we propose to apply Sequential Monte Carlo (SMC), together with a velocity-driven transition kernel. Considering discrete-time steps $0 = t_0 < \cdots < t_M = 1$, The key ingredients of SMC are proposals $\{\mathcal{F}_{t_m}(x_{t_{m+1}}|x_{t_m})\}_{m=0}^{M-1}$ and weighting functions $\{w_{t_m}\}_{m=0}^{M}$. At the initial step, one draws K particles of $x_{t_0}^{(k)} \sim p_{t_0}$ and set $w_{t_0}^{(k)} = p_{t_0}(x_{t_0}^{(k)})$, and sequentially repeats the following steps for $m = 1, \ldots, M$: i) *resample* $\{x_{t_{m-1}}^{(k)}\}_{k=1}^{K} \sim$ Systematic$(\{x_{t_{m-1}}^{(k)}\}_{k=1}^{K}; \{w_{t_{m-1}}^{(k)}\}_{k=1}^{K})$; ii) *propose* $x_{t_m}^{(k)} \sim \mathcal{F}_{t_{m-1}}(x_{t_m}|x_{t_{m-1}})$ for $k = 1, \ldots, K$; and iii) *weight* $w_t^{(k)} = \tilde{p}_{t_m}(x_{t_m}^{(k)})/\tilde{p}_{t_{m-1}}(x_{t_m}^{(k)})$ for $k = 1, \ldots, K$. The time derivatives at time step t can therefore be estimated via $\partial_t \log Z_t \approx \sum_k \frac{w_t^{(k)}}{\sum_k w_t^{(k)}} \partial_t \log \tilde{p}_t(x_t^{(k)})$ (see Appendix A.2 for details). This approximation becomes arbitrarily accurate in the limit as much as particles are used (Chopin et al., 2020, Proposition 11.4).

The choice of the proposal in SMC is critical as a poor proposal can lead to trajectories that quickly collapse onto a single ancestor, reducing particle diversity and effectiveness. To address this issue, we propose incorporating Markov Chain Monte Carlo (MCMC) steps into the SMC framework (Van Der Merwe et al., 2000). To further enhance MCMC convergence, we integrate it with a trainable velocity model. Specifically, the transition kernel $\mathcal{F}_{t_m}(x_{t_{m+1}}|x_{t_m})$ comprises two steps i) *velocity move* with an Euler update $\hat{x}_{t_{m+1}} \leftarrow x_{t_m} + v_{t_m}(x_{t_m}; \theta)(t_{m+1} - t_m)$ to provide a better initialisation; and ii) *MCMC move* with a HMC (Neal, 2012) refinement $x_{t_{m+1}} \sim \text{HMC}(\hat{x}_{t_{m+1}})$ to ensure consistency with the target distribution. This velocity-driven MCMC kernel operates in a bootstrap manner, with the velocity move offering an informed initialisation that improves the efficiency of the subsequent MCMC step. As training progresses, the velocity model becomes increasingly accurate, generating high-quality proposals that reduce the corrections required during the MCMC step. This synergy between the velocity move and the MCMC refinement not only accelerates convergence but also preserves particle diversity, making the approach robust in high-dimensional or complex settings.

Further Variance Reduction with Control Variates. To further reduce the variance, a key observation is that for any given velocity v_t, the following identity holds

$$\partial_t \log Z_t = \underset{c_t}{\text{argmin}} \mathbb{E}_{p_t}(\xi_t(x; v_t) - c_t)^2, \xi_t(x; v_t) \triangleq \partial_t \log \tilde{p}_t(x) + \nabla_x \cdot v_t(x) + v_t(x) \cdot \nabla_x \log p_t(x). \quad (4)$$

See Appendix B.1 for proof. Thus, one can calculate the optimal c_t via $\partial_t \log Z_t = \mathbb{E}_{p_t} \xi_t(x; v_t)$, which can be approximated using Monte Carlo estimation. Empirically, we observe that Equation (4) achieves lower variance compared to the approximation $\partial_t \log Z_t = \mathbb{E}_{p_t} \partial_t \log \tilde{p}_t(x)$, and it sometimes leads to better optimisation. We visualize the comparison of the standard deviation of the two estimation methods in Figure 1, with the corresponding loss plots deferred to Figure 5. In Appendix B.2, we

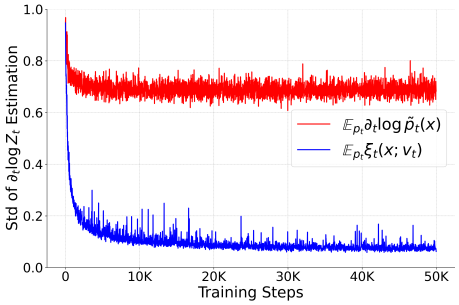
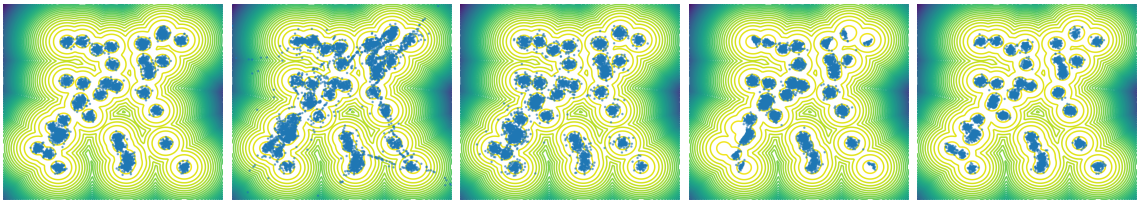


Figure 1: Standard deviation of the estimation of $\partial_t \log Z_t$.

Table 1: Comparison of neural samplers on GMM-40, MW-32, and DW-4 energy functions, with mean and standard deviation based on five evaluations using different random seeds.

Energy →	GMM-40 ($d=2$)		MW-32 ($d=32$)		DW-4 ($d=8$)		
Method ↓	$\mathcal{E}$-$\mathcal{W}_2$	$\mathcal{X}$-TV	$\mathcal{E}$-TV	$\mathcal{X}$-$\mathcal{W}_2$	$\mathcal{E}$-$\mathcal{W}_2$	$\mathcal{E}$-TV	$\mathcal{D}$-TV
FAB (Midgley et al., 2023)	8.89±2.20	0.84±0.19	0.25±0.01	5.78±0.02	0.64±0.20	0.22±0.01	0.09±0.01
iDEM (Sadegh et al., 2024)	1.27±0.21	0.83±0.01	0.63±0.15	8.18±0.04	0.19±0.05	0.21±0.01	0.10±0.01
LFIS (Tian et al., 2024)	0.27±0.21	0.84±0.01	∞	8.89±0.03	6.06±1.05	0.66±0.02	0.29±0.01
NFS^2-128 (ours)	0.46±0.14	0.67±0.00	0.16±0.00	6.17±0.01	0.44±0.03	0.10±0.01	0.07±0.01
NFS^2-64 (ours)	1.32±0.29	0.69±0.01	0.18±0.00	6.34±0.01	0.98±0.16	0.13±0.01	0.11±0.01
NFS^2-32 (ours)	4.38±1.14	0.72±0.01	0.49±0.01	9.05±0.01	14.97±0.82	0.41±0.01	0.28±0.01



Ground Truth FAB iDEM LFIS NFS^2 (ours)

Figure 2: Samples of GMM-40, with contour lines representing the ground truth distribution.

provide a control variate perspective to explain this observation. Therefore, when combined with SMC, the time derivatives can be estimated as $\partial_t \log Z_t \approx \sum_k \frac{w_t^{(k)}}{\sum_k w_t^{(k)}} \xi_t(x_t^{(k)}; v_t)$. To summarize, the velocity can then be learned by iterating the following steps i) $\theta \leftarrow \operatorname{argmin}_\theta \int_0^1 \mathbb{E}_{p_t}(\xi_t(x; v_t(x;\theta)) - c_t)^2 \,\mathrm{d}t$; ii) $c_t \leftarrow \mathbb{E}_{p_t} \xi_t(x; v_t(x;\theta))$. Notably, Máté and Fleuret (2023) propose a similar method, where c_t is parametrised as a neural network and trained using stochastic gradient descent by optimising objective in Equation (4), rather than using SMC in conjunction with control variates.

3.2. Neural Flow Shortcut Sampler

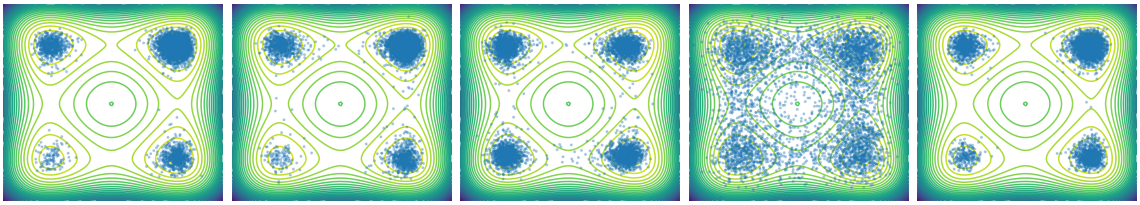
With the time derivative estimator established in the previous sections, the velocity can be learned by minimizing the loss defined in Equation (3). However, during sampling, small step sizes are required to control discretization error, resulting in high computational costs. To address this, we draw inspiration from the recent success of shortcut models (Frans et al., 2024) and introduce an additional shortcut regularization to alleviate this issue. The basic idea is to parameterise a shortcut model $s_t(x_t, d; \theta)$ to account for the future curvature, such that $x_{t+d} = x_t + s_t(x_t, d; \theta)d$. To this end, one can regularise the shortcut model to ensure that $s_t(x_t, 2d; \theta) = s_t(x_t, d; \theta)/2 + s_{t+d}(x_{t+d}, d; \theta)/2 =: s_{\text{target}}, \forall d, t$, which leads to the final objective

$$\mathcal{L}(\theta) = \mathbb{E}_{q(x),w(t)} \left[\delta_t^2(x; s_t(\cdot, 0; \theta)) + \mathbb{E}_{p(d)} \| s_t(x, 2d; \theta) - s_{\text{target}} \|_2^2 \right]. \tag{5}$$

We refer to the proposed methods as neural flow shortcut samplers (NFS^2), using 128 sampling steps by default unless specified otherwise. The training and sampling procedures are summarized in Algorithms 2 and 3, respectively.

4. Experiments

We evaluate our method against baselines on three distinct energy functions: i) a 2-dimensional Gaussian Mixture Model with 40 components (GMM-40), ii) a 32-dimensional many-well



Ground Truth FAB iDEM LFIS NFS^2 (ours)

Figure 3: 2D marginal samples from the 1st and 3rd dimensions of MW-32.

(MW-32) distribution, and iii) an 8-dimensional 4-particle double-well (DW-4) potential. Experimental details and additional results are provided in Appendices D and E.

Baselines. We compare against three recent methods: Flow Annealed Importance Sampling Bootstrap (FAB) (Midgley et al., 2023), Iterated Denoising Energy Matching (iDEM) (Sadegh et al., 2024) with 1000 sampling steps, and Liouville Flow Importance Sampling (LFIS) (Tian et al., 2024) with 256 sampling steps. We denote the proposed method with M sampling steps as NFS^2-M.

Results and Discussion. Table 1 presents a comparison of our method with baselines across various metrics, showing that NFS^2 outperforms better in most cases. Specifically, compared to LFIS, the method most similar to ours in learning velocity and estimating $\partial_t \log Z_t$ via importance sampling, NFS^2 exhibits superior performance. This highlights the effectiveness of our variance reduction techniques through SMC and control variates. When compared to iDEM, the current SOTA diffusion-based sampler, NFS^2 surpasses iDEM on GMM-40 and MW-32, achieving comparable performance on DW-4. Notably, NFS^2 achieves this with significantly fewer sampling steps than iDEM, further emphasizing its superiority. We visualize the generated GMM-40 samples in Figure 2, where NFS^2 is the only method that captures both the diversity and sharpness of the modes. MW-32 samples are shown in Figure 3, where LFIS and iDEM fail to accurately model the position and weight of modes, while FAB and NFS^2 excel at both. Moreover, we also visualize histograms for sample energy and interatomic distance in Figure 4. As shown, NFS^2 achieves competitive performance with both FAB and iDEM, and notably outperforms LFIS. For additional experimental results, please refer to Appendix E.

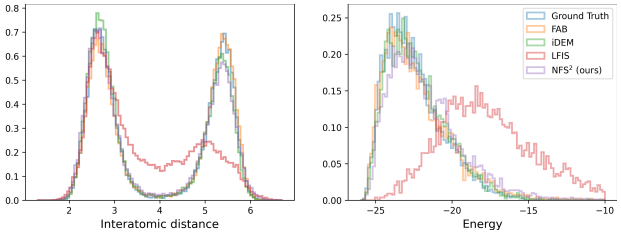


Figure 4: Histogram of interatomic distance and sample energy on DW-4.

5. Related Work

Sampling from given probability distributions has been a longstanding research challenge. Monte Carlo methods, such as Annealed Importance Sampling (Neal, 2001) and Sequential Monte Carlo (Del Moral et al., 2006), are considered the gold standards for sampling, but they tend to be computationally expensive and often suffer from slow convergence (Roberts and Rosenthal, 2001). Amortised variational methods like normalizing flows (Rezende and Mohamed, 2015) and latent variable models (He et al., 2024) provide appealing alternatives to matching the target distribution, offering faster inference but often at the cost of approximation errors and limited expressiveness. Hybrid approaches (Wu et al., 2020;

Zhang et al., 2021; Geffner and Domke, 2021; Matthews et al., 2022; Midgley et al., 2023) that combine MCMC and variational inference have shown promising potential by leveraging the strengths of both methods.

Building upon the success of generative modelling, diffusion models (Ho et al., 2020) and flow matching (Lipman et al., 2022) have been applied to sampling tasks. Specifically, Vargas et al. (2023); Nusken et al. (2024) exploit diffusion processes for learning to sample. However, these approaches require simulation to compute the objective. To resolve this issue, Sadegh et al. (2024) propose to use a bi-level training scheme that iteratively generates samples and performs score matching with Monte Carlo estimates of the target, resembling training diffusion models, and does not require simulation. OuYang et al. (2024) introduce a variant that replaces score matching with direct regression on the energy, which is shown to reduce variance. Similarly, Woo and Ahn (2024) present another variant that targets on the MC-estimated vector fields in a flow matching framework. Other approaches also focus on learning the velocity field; for instance, Tian et al. (2024); Máté and Fleuret (2023) learn the velocity field to satisfy the continuity equation of the given probability path.

Beyond the above methods, stochastic control (Pavon, 1989; Tzen and Raginsky, 2019) has also been applied to the sampling. For example, Zhang and Chen (2021) propose path integral sampler (PIS) based on the connections between sampling and optimal control (Chen et al., 2016). Berner et al. (2022) also establish the connection between optimal control and generative modelling based on stochastic differential equations (Kloeden et al., 1992), which can be applied in sampling. Generative flow networks (GFlowNets) (Lahlou et al., 2023) are appealing alternatives for sampling from unnormalised densities. Zhang et al. (2022) establishes a connection between diffusion models and GFlowNets, leveraging this relationship to enhance learning-based sampling (Zhang et al., 2023), which only requires simulating partial trajectories, improving the efficiency compared to PIS.

6. Conclusions and Limitations

In this paper, we proposed neural flow shortcut samplers (NFS^2), a flow-based sampler that learns a velocity field to satisfy the continuity equation. To estimate the intractable time derivative of the log-partition function in the continuity equation, we introduced velocity-driven Sequential Monte Carlo with control variates, effectively reducing estimation variance. Our method achieves competitive or superior performance compared to SOTA approaches like FAB and iDEM, while outperforming flow-based samples like LFIS, which suffers from high variance due to the utilisation of importance sampling.

A key challenge of flow-based samplers is the computation of the divergence (see Equation (3)), which becomes prohibitive in high-dimensional settings. While the Hutchinson estimator (Hutchinson, 1989) can be used in practice, it introduces both variance and bias. Alternatively, more advanced architectures can be employed where the divergence is computed analytically (Gerdes et al., 2023). By adopting such architectures, we expect our approach to be scalable to more complex applications, such as molecular simulation (Frenkel and Smit, 2023), Lennard-Jones potential (Klein et al., 2024), and Bayesian inference (Neal, 1993). Moreover, while the shortcut model reduces the number of sampling steps required, achieving exact likelihood estimation within this framework remains unclear, presenting a promising direction for future research.

References

Michael S Albergo, Gurtej Kanwar, and Phiala E Shanahan. Flow-based generative models for markov chain monte carlo in lattice field theory. *Physical Review D*, 100(3):034515, 2019.

Julius Berner, Lorenz Richter, and Karen Ullrich. An optimal control perspective on diffusion-based generative modeling. *arXiv preprint arXiv:2211.01364*, 2022.

Yongxin Chen, Tryphon T Georgiou, and Michele Pavon. On the relation between optimal transport and schrödinger bridges: A stochastic control viewpoint. *Journal of Optimization Theory and Applications*, 169:671–691, 2016.

Nicolas Chopin, Omiros Papaspiliopoulos, et al. *An introduction to sequential Monte Carlo*, volume 4. Springer, 2020.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006.

Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. *Sequential Monte Carlo methods in practice*, pages 3–14, 2001.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.

Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*. Elsevier, 2023.

Tomas Geffner and Justin Domke. Using large ensembles of control variates for variational inference. *Advances in Neural Information Processing Systems*, 31, 2018.

Tomas Geffner and Justin Domke. Mcmc variational inference via uncorrected hamiltonian annealing. *Advances in Neural Information Processing Systems*, 34:639–651, 2021.

Mathis Gerdes, Pim de Haan, Corrado Rainone, Roberto Bondesan, and Miranda CN Cheng. Learning lattice quantum field theories with equivariant continuous flows. *SciPost Physics*, 15(6):238, 2023.

Jiajun He, Wenlin Chen, Mingtian Zhang, David Barber, and José Miguel Hernández-Lobato. Training neural samplers with reverse diffusive kl divergence. *arXiv preprint arXiv:2410.12456*, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Herman Kahn. Random sampling (monte carlo) techniques in neutron attenuation problems. i. *Nucleonics (US) Ceased publication*, 6(See also NSA 3-990), 1950.

Leon Klein, Andreas Krämer, and Frank Noé. Equivariant flow matching. *Advances in Neural Information Processing Systems*, 36, 2024.

Peter E Kloeden, Eckhard Platen, Peter E Kloeden, and Eckhard Platen. *Stochastic differential equations*. Springer, 1992.

Jonas Köhler, Leon Klein, and Frank Noe. Equivariant flows: Exact likelihood generative learning for symmetric densities. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5361–5370. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/kohler20a.html.

R Komanduri, N Chandrasekaran, and LM Raff. Md simulation of nanometric cutting of single crystal aluminum–effect of crystal orientation and direction of cutting. *Wear*, 242 (1-2):60–88, 2000.

Salem Lahlou, Tristan Deleu, Pablo Lemos, Dinghuai Zhang, Alexandra Volokhova, Alex Hernández-Garcıa, Léna Néhale Ezzine, Yoshua Bengio, and Nikolay Malkin. A theory of continuous generative flow networks. In *International Conference on Machine Learning*, pages 18269–18300. PMLR, 2023.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Hao Liu, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu. Action-depedent control variates for policy optimization via stein's identity. *arXiv preprint arXiv:1710.11198*, 2017.

Bálint Máté and François Fleuret. Learning interpolations between boltzmann densities. *arXiv preprint arXiv:2301.07388*, 2023.

Alex Matthews, Michael Arbel, Danilo Jimenez Rezende, and Arnaud Doucet. Continual repeated annealed flow transport monte carlo. In *International Conference on Machine Learning*, pages 15196–15219. PMLR, 2022.

Laurence Illing Midgley, Vincent Stimper, Gregor NC Simm, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Flow annealed importance sampling bootstrap. *International Conference on Learning Representations (ICLR)*, 2023.

Radford M Neal. Probabilistic inference using markov chain monte carlo methods. 1993.

Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11:125–139, 2001.

Radford M Neal. Mcmc using hamiltonian dynamics. *arXiv preprint arXiv:1206.1901*, 2012.

Nikolas Nusken, Francisco Vargas, Shreyas Padhy, and Denis Blessing. Transport meets variational inference: Controlled monte carlo diffusions. In *The Twelfth International Conference on Learning Representations: ICLR 2024*, 2024.

RuiKang OuYang, Bo Qiang, and José Miguel Hernández-Lobato. Bnem: A boltzmann sampler based on bootstrapped noised energy matching. *arXiv preprint arXiv:2409.09787*, 2024.

Michele Pavon. Stochastic control and nonequilibrium thermodynamical systems. *Applied Mathematics and Optimization*, 19:187–202, 1989.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial intelligence and statistics*, pages 814–822. PMLR, 2014.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling for various metropolis-hastings algorithms. *Statistical science*, 16(4):351–367, 2001.

Tara Akhound Sadegh, Jarrid Rector-Brooks, Avishek Joey Bose, Sarthak Mittal, Pablo Lemos, Cheng-Hao Liu, Marcin Sendera, Siamak Ravanbakhsh, Gauthier Gidel, Yoshua Bengio, Nikolay Malkin, and Alexander Tong. Iterated denoising energy matching for sampling from boltzmann densities. *ArXiv*, abs/2402.06121, 2024. URL https://api.semanticscholar.org/CorpusID:267617166.

Charles Stein, Persi Diaconis, Susan Holmes, and Gesine Reinert. Use of exchangeable pairs in the analysis of simulations. *Lecture Notes-Monograph Series*, pages 1–26, 2004.

H Jean Thiébaux and Francis W Zwiers. The interpretation and estimation of effective sample size. *Journal of Applied Meteorology and Climatology*, 23(5):800–811, 1984.

Yifeng Tian, Nishant Panda, and Yen Ting Lin. Liouville flow importance sampler. *arXiv preprint arXiv:2405.06672*, 2024.

Belinda Tzen and Maxim Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. In *Conference on Learning Theory*, pages 3084–3114. PMLR, 2019.

Rudolph Van Der Merwe, Arnaud Doucet, Nando De Freitas, and Eric Wan. The unscented particle filter. *Advances in neural information processing systems*, 13, 2000.

Francisco Vargas, Will Grathwohl, and Arnaud Doucet. Denoising diffusion samplers. *arXiv preprint arXiv:2302.13834*, 2023.

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

Dongyeop Woo and Sungsoo Ahn. Iterated energy-based flow matching for sampling from boltzmann densities. *arXiv preprint arXiv:2408.16249*, 2024.

Hao Wu, Jonas Köhler, and Frank Noé. Stochastic normalizing flows. *Advances in Neural Information Processing Systems*, 33:5933–5944, 2020.

Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. Mars: Markov molecular sampling for multi-objective drug discovery. *arXiv preprint arXiv:2103.10432*, 2021.

Dinghuai Zhang, Ricky TQ Chen, Nikolay Malkin, and Yoshua Bengio. Unifying generative models with gflownets and beyond. *arXiv preprint arXiv:2209.02606*, 2022.

Dinghuai Zhang, Ricky TQ Chen, Cheng-Hao Liu, Aaron Courville, and Yoshua Bengio. Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. *arXiv preprint arXiv:2310.02679*, 2023.

Guodong Zhang, Kyle Hsu, Jianing Li, Chelsea Finn, and Roger B Grosse. Differentiable annealed importance sampling and the perils of gradient noise. *Advances in Neural Information Processing Systems*, 34:19398–19410, 2021.

Qinsheng Zhang and Yongxin Chen. Path integral sampler: a stochastic control approach for sampling. *arXiv preprint arXiv:2111.15141*, 2021.

Appendix for "Neural Flow Samplers with Shortcut Models"

Contents

Appendix A. Importance Sampling and Sequential Monte Carlo

This section reviews the basic Sequential Monte Carlo (SMC) algorithm. We begin by introducing importance sampling and its application to estimating the intractable time derivative $\partial_t \log Z_t$, as presented in Tian et al. (2024). We then proceed with an introduction to Sequential Monte Carlo, which is employed in our methods to estimate $\partial_t \log Z_t$.

A.1. Importance Sampling

Consider a target distribution $\pi(x) = \frac{\rho(x)}{Z}$, where $\rho(x) \geq 0$ is the unnormalised probability density and $Z = \int \rho(x)\,\mathrm{d}x$ denotes the normalising constant, which is typically intractable. For a test function $\phi(x)$ of interest, estimating its expectation under π through direct sampling can be challenging. Importance sampling (IS) (Kahn, 1950) instead introduces a proposal distribution q, which is easy to sample from, and proposes an expectation estimator as follows

$$\mathbb{E}_{\pi(x)}[\phi(x)] = \frac{1}{Z}\mathbb{E}_{q(x)}\left[\frac{\rho(x)}{q(x)}\phi(x)\right] = \frac{\mathbb{E}_{q(x)}\left[\frac{\rho(x)}{q(x)}\phi(x)\right]}{\mathbb{E}_{q(x)}\left[\frac{\rho(x)}{q(x)}\right]}. \tag{6}$$

Thus, the expectation can be estimated via the Monte Carlo method

$$\mathbb{E}_{\pi(x)}[\phi(x)] \approx \sum_{k=1}^{K} \frac{w^{(k)}}{\sum_{=1}^{N} w^{(j)}}\phi(x^{(k)}), \quad x^{(k)} \sim q(x), \tag{7}$$

where $w^{(k)} = \frac{\rho(x^{(k)})}{q(x^{(k)})}$ denotes the importance weight. While importance sampling yields a consistent estimator as $N \to \infty$, it typically suffers from high variance and low effective sample size (Thiébaux and Zwiers, 1984) when the proposal deviates from the target distribution. In theory, a zero-variance estimator can be achieved if $q(x) \propto \rho(x)\phi(x)$; however, this condition is rarely satisfied in practice. This limitation renders importance sampling inefficient in high-dimensional spaces, as a large number of Monte Carlo samples are required to mitigate the variance.

Approximating $\partial_t \log Z_t$ with Importance Sampling. Tian et al. (2024) propose approximating $\partial_t \log Z_t$ using importance sampling, where they express $\partial_t \log Z_t \approx \sum_k \frac{w_t^{(k)}}{\sum_k w_t^{(k)}} \partial_t \log \tilde{p}_t(x_t^{(k)})$. Here $x_t^{(k)} \sim p_t(x;\theta)$ denotes the sample generated by the velocity $v_t(x;\theta)$ at time t, and $\log w_t^{(k)} = \int_0^t \delta_\tau(x_\tau; v_t(\cdot;\theta))\,\mathrm{d}\tau$. For completeness, we provide a step-by-step recall of the proof of the correctness of this estimator from Tian et al. (2024).

First, we show that $\partial_t \log Z_t$ is given by the expectation over p_t:

$$\partial_t \log Z_t = \partial_t \log \int \tilde{p}_t(x)\,\mathrm{d}x = \frac{1}{Z_t}\int \tilde{p}_t(x)\partial_t \log \tilde{p}_t(x)\,\mathrm{d}x = \mathbb{E}_{p_t(x)}[\partial_t \log \tilde{p}_t(x)]. \tag{8}$$

$\partial_t \log Z_t$ therefore can be estimated via importance sampling

$$\partial_t \log Z_t \approx \sum_{k=1}^{K} \frac{w_t^{(k)}}{\sum_j w_t^{(j)}} \partial_t \log \tilde{p}_t(x^{(k)}), \quad x^{(k)} \sim p_t(x;\theta), \tag{9}$$

where $w_t^{(k)} = \frac{p_t(x^{(k)})}{\tilde{p}_t(x^{(k)};\theta)}$. Next, we show that the weight $\log w_t^{(k)}$ is given by integrating $\delta_t(x; v_t(\cdot;\theta))$ on $[0,t]$, where δ_t is defined in Equation (3) i.e. $\log \frac{p_t(x_t)}{\tilde{p}_t(x_t;\theta)} = \int_0^t \delta_s(x; v_s(\cdot;\theta))\,\mathrm{d}s$.

We begin by computing the instantaneous rate of change of the log-densities of the model $p_t(x_t;\theta)$ along the trajectories generated by $v_t(x_t;\theta)$, as in Equation (2)

$$\begin{aligned}
\partial_t[\log p_t(x_t;\theta)] &= \partial_t \log p_t(x_t;\theta) + \nabla_{x_t} \log p_t(x_t;\theta) \cdot \frac{\mathrm{d}x_t}{\mathrm{d}t} \\
&= (-\nabla_{x_t} \cdot v_t(x_t;\theta) - v_t(x_t;\theta) \cdot \nabla_{x_t} \log p_t(x_t;\theta)) + \nabla_{x_t} \log p_t(x_t;\theta) \cdot v_t(x_t;\theta) \\
&= -\nabla_{x_t} \cdot v_t(x_t;\theta).
\end{aligned} \tag{10}$$

Similarly, the instantaneous rate of change of the log-densities of the target $p_t(x_t)$ along the trajectories generated by $v_t(x_t;\theta)$ is

$$\begin{aligned}
\partial_t[\log p_t(x_t)] &= \partial_t \log p_t(x_t) + \nabla_{x_t} \log p_t(x_t) \cdot \frac{\mathrm{d}x_t}{\mathrm{d}t} \\
&= \partial_t \log p_t(x_t) + \nabla_{x_t} \log p_t(x_t) \cdot v_t(x_t;\theta).
\end{aligned} \tag{11}$$

Note that the score of the target is tractable $\nabla_x \log p_t(x) = \nabla_x \log \tilde{p}_t(x)$. Therefore, the log densities along the trajectories can be computed via

$$\log p_t(x_t;\theta) = \log p_0(x_0;\theta) - \int_0^t \nabla_{x_s} \cdot v_s(x_s;\theta)\,\mathrm{d}s \tag{12}$$

$$\log p_t(x_t) = \log p_0(x_0) + \int_0^t [\partial_s \log p_s(x_s) + \nabla_{x_s} \log p_s(x_s) \cdot v_s(x_s;\theta)]\,\mathrm{d}s. \tag{13}$$

Since $p_0(x;\theta) = p_0(x) = \eta(x)$ due to the annealing path construction $p_t \propto \rho^t \eta^{1-t}$, we have

$$\begin{aligned}\log \frac{p_t(x_t)}{p_t(x_t;\theta)} &= \int_0^t \left[\nabla_{x_s} \cdot v_s(x_s;\theta) + \partial_s \log p_s(x_s) + \nabla_{x_s} \log p_s(x_s) \cdot v_s(x_s;\theta)\right] \mathrm{d}s \\ &= \int_0^t \delta_s(x; v_s(\cdot;\theta))\, \mathrm{d}s, \end{aligned} \tag{14}$$

which completes the proof.

Remark. Although importance sampling offers an elegant method for estimating the intractable time derivatives $\partial_t \log Z_t$, it faces two main challenges. First, as previously discussed, importance sampling typically suffers from high variance and requires large sample sizes to improve the effective sample size. More critically, the computation of the weight involves the intractable term $\partial_t \log p_s(x_t)$, which in turn depends on $\partial_t \log Z_t = \mathbb{E}_{p_t(x)}[\partial_t \log \tilde{p}_t(x)]$. Tian et al. (2024) approximate this expectation naively by averaging $\partial_t \log \tilde{p}_t(x)$ over the mini-batch during training, which introduces both approximation errors and bias in the importance weights.

In the following section, we introduce Sequential Monte Carlo, which is employed in our methodology to estimate $\partial_t \log Z_t$, balancing the efficiency of the short-run MCMC driven by the velocity with the effectiveness of low variance.

A.2. Sequential Monte Carlo

Sequential Monte Carlo (SMC) provides an alternative method to estimate the intractable expectation $\mathbb{E}_{p_t(x)}[\phi(x)]$. Specifically, SMC decomposes the task into easier subproblems involving a set of unnormalised intermediate target distributions $\{\tilde{p}_{t_m}(x_{t_m})\}_{m=0}^M$.[1] We begin by introducing sequential importance sampling (SIS):

$$\begin{aligned}\mathbb{E}_{p_{t_m}(x)}[\phi(x)] &= \int q(x_{t_0:t_m}) \frac{p(x_{t_0:t_m})}{q(x_{t_0:t_m})} \phi(x_{t_m})\, \mathrm{d}x_{t_0:t_m} \\ &\approx \frac{1}{K} \sum_{k=1}^K \frac{p(x^{(k)}_{t_0:t_m})}{q(x^{(k)}_{t_0:t_m})} \phi(x^{(k)}_{t_m}), \quad \text{where} \quad x^{(k)}_{t_0:t_m} \sim q(x^{(k)}_{t_0:t_m}). \end{aligned} \tag{15}$$

The importance weights are $w^{(k)}_{t_m} = \frac{p(x^{(k)}_{t_0:t_m})}{q(x^{(k)}_{t_0:t_m})}$. The key ingredients of SMC are the proposal distributions $q(x_{t_0:t_m})$ and the target distributions $p(x_{t_0:t_m})$. Here we consider a general form associated with a sequence of forward kernels $q(x_{t_0:t_m}) = q(x_{t_0}) \prod_{s=0}^{m-1} \mathcal{F}_{t_s}(x_{t_{s+1}}|x_{t_s})$, and the target distribution is defined by a sequence of backward kernels $p(x_{t_0:t_m}) = p(x_{t_m}) \prod_{s=0}^{m-1} \mathcal{B}_{t_s}(x_{t_s}|x_{t_{s+1}})$. Substituting this into the expression for the importance weights gives

$$w^{(k)}_{t_m} = \frac{p(x^{(k)}_{t_0:t_m})}{q(x^{(k)}_{t_0:t_m})} = \frac{p(x_{t_m}) \prod_{s=0}^{m-1} \mathcal{B}_{t_s}(x_{t_s}|x_{t_{s+1}})}{q(x_{t_0}) \prod_{s=0}^{m-1} \mathcal{F}_{t_s}(x_{t_{s+1}}|x_{t_s})} = w^{(k)}_{t_{m-1}} W^{(k)}_{t_m}, \tag{16}$$

1. We consider a discrete-time schedule that satisfies $0 = t_0 < t_1 < \cdots < t_m < \cdots < t_{M-1} < t_M = 1$.

where $W_{t_m}^{(k)}$, termed the incremental weights, are calculated as,

$$W_{t_m}^{(k)} = \frac{p_{t_m}(x_{t_m})}{p_{t_{m-1}}(x_{t_{m-1}})} \frac{\mathcal{B}_{t_m}(x_{t_{m-1}}|x_{t_m})}{\mathcal{F}_{t_m}(x_{t_m}|x_{t_{m-1}})}. \tag{17}$$

By defining the backward kernel as $\mathcal{B}_{t_m}(x_{t_{m-1}}|x_{t_m}) = \frac{p_{t_{m-1}}(x_{t_{m-1}})\mathcal{F}_{t_m}(x_{t_m}|x_{t_{m-1}})}{p_{t_{m-1}}(x_{t_m})}$, the incremental weight is tractable and becomes

$$W_{t_m}^{(k)} = \frac{p_{t_m}(x_{t_m})}{p_{t_{m-1}}(x_{t_m})}. \tag{18}$$

Therefore, the expectation can be approximated as

$$\mathbb{E}_{p_{t_m}(x)}[\phi(x)] \approx \sum_k \tilde{w}_{t_m}^{(k)} \phi(x_{t_m}), \quad \tilde{w}_{t_m}^{(k)} = \frac{w_{t_m}^{(k)}}{\sum_j w_{t_m}^{(j)}}, \quad w_{t_m}^{(k)} = w_{t_{m-1}}^{(k)} W_{t_m}^{(k)} \propto w_{t_{m-1}}^{(k)} \frac{\tilde{p}_{t_m}(x_t)}{\tilde{p}_{t_{m-1}}(x_{t_m})}.$$

The SIS method is elegant, as the weights can be computed on the fly. However, with a straightforward application of SIS, the distribution of importance weights typically becomes increasingly skewed as t progresses, resulting in many samples having negligible weights. This imbalance reduces the effective sample size and the overall efficiency of the algorithm. To alleviate this issue, a common approach used in SMC is to introduce a resampling step. At each time step t, the samples $x_t^{(k)}$ are resampled using systematic resampling based on the normalized weights $\tilde{w}_t^{(k)}$[2]. The resampled particles are then assigned equal weights to mitigate the bias introduced by the skewness in the weight distribution. This resampling trick prevents the sample set from degenerating, where only a few particles carry significant weight while others contribute minimally. By periodically resampling, the algorithm maintains diversity in the particle set. It ensures that the estimation is focused

Algorithm 1 Velocity-Driven SMC

Input: velocity $v_t(\cdot;\theta)$; # sample K; time steps $\{t_m\}_{m=0}^M$

Output: samples and weights $\{x_{t_m}^{(k)}, \tilde{w}_{t_m}^{(k)}\}_{k=1,m=0}^{K,M}$

procedure VD-SMC($v_t, K, \{t_m\}_{m=0}^M$)
  for $k = 1, \dots, K$ **do**
    $x_0^{(k)} \sim p_0(x_0), \quad w_0^{(k)} = p_0(x_0^{(k)})$
  end for
  $\tilde{w}_0^{(k)} = w_0^{(k)} / \sum_{i=1}^K w_0^{(i)}$
  for $m = 1, \dots, M$ **do**
    for $k = 1, \dots, K$ **do**
      if $\mathrm{ESS}(\tilde{w}_{t_{m-1}}^{1:K}) < \mathrm{ESS}_{\min}$ **then**
        $a_{t_m}^{(k)} \sim \mathrm{Systematic}(\tilde{w}_{t_{m-1}}^{1:K})$, $\hat{w}_{t_{m-1}}^{(k)} = 1$
      else
        $a_{t_m}^{(k)} = k, \quad \hat{w}_{t_{m-1}}^{(k)} = w_{t_{m-1}}^{(k)}$
      end if
      $\mathrm{d}t \leftarrow t_m - t_{m-1}$
      $x_{t_m}^{(k)} = \mathrm{HMC}(x_{t_{m-1}}^{(a_{t_m}^{(k)})} + v_{t_{m-1}}(x_{t_{m-1}}^{(a_{t_m}^{(k)})}; \theta)\, \mathrm{d}t)$
      $w_{t_m}^{(k)} = \hat{w}_{t_{m-1}}^{(k)} \frac{\tilde{p}_{t_m}(x_{t_m}^{(k)})}{\tilde{p}_{t_{m-1}}(x_{t_m}^{(k)})}$
    end for
    $\tilde{w}_{t_m} = w_{t_m}^{(k)} / \sum_{i=1}^K w_{t_m}^{(i)}$
  end for
end procedure

2. Rather than resampling at every time step t, a more advanced resampling method involves making the resampling decision adaptively, based on criteria such as the Effective Sample Size (Doucet et al., 2001).

on regions of high probability density, leading to less skewed importance weight distributions. To encourage the convergence of MCMC transition kernels, we also introduce a velocity-driven step. The implementation of the proposed velocity-driven sequential Monte Carlo (VD-SMC) is given by Algorithm 1. Given the sample size K and the time schedule $\{t_m\}_{m=1}^M$ with $t_0 = 0, t_M = 1$, the algorithm VD-SMC returns the samples and importance weights $\{x_{t_m}^{(k)}, \tilde{w}_{t_m}^{(k)}\}_{k=1,m=0}^{K,M}$. Therefore, the intractable time derivative can be approximated as $\partial_t \log Z_t = \mathbb{E}_{p_t} \partial_t \log \tilde{p}_t(x) \approx \sum_k \tilde{w}_t^{(k)} \partial_t \log \tilde{p}_t(x_t^{(k)})$. However, as illustrated in Figure 1, the estimation of $\mathbb{E}_{p_t} \partial_t \log \tilde{p}_t(x)$ exhibits higher variance compared to using $\mathbb{E}_{p_t} \xi_t(x; v_t)$. Therefore, in practice, we approximate the time derivative as $\partial_t \log Z_t = \mathbb{E}_{p_t} \xi_t(x; v_t(\cdot; \theta_{\text{sg}})) \approx \sum_k \tilde{w}_t^{(k)} \xi_t(x_t^{(k)}; v_t(\cdot; \theta_{\text{sg}}))$, where θ_{sg} denotes the parameters with gradients detached.

Appendix B. Variance Reduction with Control Variates

B.1. Proof of Equation (4)

Recall that in Equation (4), we show that the following equation holds:

$$\partial_t \log Z_t = \operatorname*{argmin}_{c_t} \mathbb{E}_{p_t} (\xi_t(x; v_t) - c_t)^2, \xi_t(x; v_t) \triangleq \partial_t \log \tilde{p}_t(x) + \nabla_x \cdot v_t(x) + v_t(x) \cdot \nabla_x \log p_t(x).$$

We provide a detailed proof of this result here. First, we have the following Lemmas.

Lemma 1 (Stein's Identity (Stein et al., 2004)) *Assuming that the target density p_t vanishes at infinity, i.e., $p_t(x) = 0$, whenever $\exists d$ such that $x[d] = \infty$, where $x[d]$ denotes the d-th element of the vector x. Under this assumption, we have the result: $\int [\nabla_x \cdot v_t(x) + v_t(x) \cdot \nabla_x \log p_t(x)] \tilde{p}_t(x) \, \mathrm{d}x = 0$.*

Proof To prove the result, notice that

$$\begin{aligned}
\int [\nabla_x \cdot v_t(x) + v_t(x) \cdot \nabla_x \log p_t(x)] \tilde{p}_t(x) \, \mathrm{d}x &= \int \tilde{p}_t(x) \nabla_x \cdot v_t(x) + v_t(x) \cdot \nabla_x \tilde{p}_t(x) \, \mathrm{d}x \\
&= \int \nabla_x \cdot [v_t(x) \tilde{p}_t(x)] \, \mathrm{d}x \\
&= \sum_d \int \frac{\mathrm{d}}{\mathrm{d}x_d} [v_t(x) \tilde{p}_t(x)][d] \, \mathrm{d}x_d \\
&= \sum_d [v_t(x) \tilde{p}_t(x)][d]|_{x_d = -infty}^{x_d = +\infty} = 0,
\end{aligned}$$

where the last row applies the divergence theorem $\int_a^b f'(t) \, \mathrm{d}t = f(b) - f(a)$. ■

Lemma 2 *Let $c_t^* = \operatorname{argmin}_{c_t} \mathbb{E}_{p_t} (\xi_t(x; v_t) - c_t)^2$, then $c_t^* = \mathbb{E}_{p_t} \xi_t(x; v_t)$.*

Proof To see this, we can expand the objective

$$\mathcal{L}(c_t) = \mathbb{E}_{p_t} (\xi_t(x; v_t) - c_t)^2 = c_t^2 - 2 c_t \mathbb{E}_{p_t} \xi_t(x; v_t) + \mathrm{c} = (c_t - \mathbb{E}_{p_t} \xi_t(x; v_t))^2 + \mathrm{c}',$$

where c, c' are constants w.r.t. c_t. Therefore $c_t^* = \operatorname{argmin}_{c_t} \mathbb{E}_{p_t}(\xi_t(x; v_t) - c_t)^2 = \mathbb{E}_{p_t}\xi_t(x; v_t)$. ■

Now, it is ready to prove Equation (4). Specifically,

$$\begin{aligned} c_t^* &= \mathbb{E}_{p_t}\xi_t(x; v_t) \\ &= \frac{1}{\int \tilde{p}_t(x)\,\mathrm{d}x} \int \tilde{p}_t(x)[\partial_t \log \tilde{p}_t(x) + \nabla_x \cdot v_t(x) + v_t(x) \cdot \nabla_x \log p_t(x)]\,\mathrm{d}x \\ &= \frac{1}{\int \tilde{p}_t(x)\,\mathrm{d}x} \int \partial_t \tilde{p}_t(x) + [\nabla_x \cdot v_t(x) + v_t(x) \cdot \nabla_x \log p_t(x)]\tilde{p}_t(x)\,\mathrm{d}x \\ &= \frac{1}{\int \tilde{p}_t(x)\,\mathrm{d}x} \int \partial_t \tilde{p}_t(x)\,\mathrm{d}x \\ &= \partial_t \log Z_t, \end{aligned}$$

where the first and fourth equations follow Theorems 1 and 2, respectively, which completes the proof.

Remark. Equation (4) provides an alternative approach to estimate $\partial_t \log Z_t$. As illustrated in Figure 1, this estimation exhibits lower variance compared to using $\mathbb{E}_{p_t}\partial_t \log \tilde{p}_t(x)$. This reduction in variance can potentially lead to better optimisation. To evaluate this, we conducted experiments on GMM datasets by minimizing the loss in Equation (3), employing two different methods to estimate $\partial_t \log Z_t$: $\mathbb{E}_{p_t}\partial_t \log \tilde{p}_t(x)$ and $\mathbb{E}_{p_t}\xi_t(x; v_t)$. The loss values during training are plotted against the training steps in Figure 5. The results show that the estimator of $\mathbb{E}_{p_t}\xi_t(x; v_t)$ achieves lower loss values, highlighting the superior training effects achieved with the lower variance estimation of $\partial_t \log Z_t$.

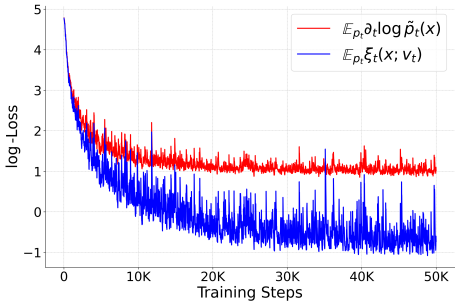


Figure 5: Training loss of using different estimators of $\partial_t \log Z_t$.

B.2. Stein Control Variates

In this section, we provide a perspective from control variates to explain the observation of variance reduction in Figure 1. In particular, consider a Monte Carlo integration problem $\mu = \mathbb{E}_\pi[f(x)]$, which can be estimated as $\hat{\mu} = \frac{1}{K}\sum_{k=1}^K f(x^{(k)}), x^{(k)} \sim \pi$. Assuming another function exists with a known mean $\gamma = \mathbb{E}_\pi[g(x)]$, we call g the control variate. We then can construct another estimator $\check{\mu} = \frac{1}{K}\sum_{k=1}^K (f(x^{(k)}) - \beta g(x^{(k)})) + \beta\gamma$, where β is a scalar coefficient and controls the scale of the control variate. It is obvious that $\mathbb{E}[\check{\mu}] = \mathbb{E}[\hat{\mu}] = \mu, \forall \beta \in \mathbb{R}$. Moreover, we can choose a β to minimize the variance of $\check{\mu}$. To obtain it, we first derive the variance of $\check{\mu}$

$$\mathbb{V}[\check{\mu}] = \frac{1}{K}(\mathbb{V}[f] - 2\beta \mathrm{Cov}(f, g) + \beta^2 \mathbb{V}[g]). \tag{19}$$

Since $\mathbb{V}[\check{\mu}]$ is convex w.r.t. β, by differentiating it w.r.t. β and zeroing it, we find the optimal value, $\beta^* = \mathrm{Cov}(f, g)/\mathbb{V}[g]$. Substituting it into Equation (19), we get the minimal variance

$$\mathbb{V}[\check{\mu}] = \frac{1}{K}\mathbb{V}[\hat{\mu}](1 - \mathrm{Corr}(f, g)^2). \tag{20}$$

Algorithm 2 Training Procedure of NFS^2 (one training epoch only for illustration)

Input: initial shortcut model $s_t(\cdot,\cdot;\theta)$, time spans $\{t_m\}_{m=0}^M$, shortcut distances $\{2^{-e}\}_{e=0}^E$
Output: trained shortcut model $s_t(\cdot,\cdot;\theta)$

$\tilde{t}_0 \leftarrow 0, \tilde{t}_m \sim \mathcal{U}([t_{m-1}, t_m]), m = 1, \dots, M$ ▷ Sample time steps
$\{x_{\tilde{t}_m}^{(k)}, \tilde{w}_{\tilde{t}_m}^{(k)}\}_{k=1,m=0}^{K,M} \leftarrow \text{VD} - \text{SMC}(s_t(\cdot,0;\theta), K, \{\tilde{t}_m\}_{m=0}^M)$ ▷ Generate samples using Alg. 1
for $t, \{x_t, \tilde{w}_t\} \sim \{x_{\tilde{t}_m}^{(k)}, \tilde{w}_{\tilde{t}_m}^{(k)}\}_{k=1,m=0}^{K,M}$ **do** ▷ Executed with mini-batch in parallel
  $\xi_t \leftarrow \partial_t \log \tilde{p}_t(x_t) + \nabla_x \cdot s_t(x_t, 0;\theta) + s_t(x_t, 0;\theta) \cdot \nabla_x \log p_t(x_t)$
  $c_t \leftarrow \sum_k \tilde{w}_t^{(k)} \left[\partial_t \log \tilde{p}_t(x_t^{(k)}) + \nabla_x \cdot s_t(x_t^{(k)}, 0; \theta_{\text{sg}}) + s_t(x_t^{(k)}, 0; \theta_{\text{sg}}) \cdot \nabla_x \log p_t(x_t^{(k)})\right]$ ▷ $\partial_t \log Z_t$
  $d \sim \mathcal{U}(\{2^{-e}\}_{e=0}^E)$ ▷ Sample shortcut distance
  $s_{\text{target}} \leftarrow s_t(x_t, d;\theta)/2 + s_{t+d}(x_{t+d}, d;\theta)/2$ ▷ Compute shortcut target
  $\mathcal{L}(\theta) \leftarrow (\xi_t - c_t)^2 + \|s_t(x_t.2d;\theta) - s_{\text{target}}\|_2^2$ ▷ Compute training loss
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$ ▷ Perform gradient update
end for

Algorithm 3 Sampling Procedure of NFS^2

Input: trained shortcut model $s_t(\cdot,\cdot;\theta)$, initial density p_0, # steps M
Output: generated samples x

$x_0 \sim p_0, \quad d \leftarrow \frac{1}{M}, \quad t \leftarrow 0$ ▷ Initialisation
for $m = 0, \dots, M-1$ **do**
  $x \leftarrow x + s_t(x, d;\theta) d$
  $t \leftarrow t + d$
end for

This shows that, given the optimal value β^*, any function g that correlates to f, whether positively or negatively, reduces the variance of the estimator, i.e., $\mathbb{V}[\check{\mu}] < \mathbb{V}[\hat{\mu}]$. In practice, the optimal β^* can be estimated from a small number of samples (Ranganath et al., 2014). However, the primary challenge lies in finding an appropriate function g. For a detailed discussion on control variates, see Geffner and Domke (2018).

Fortunately, Theorem 1 offers a systematic way to construct a control variate for $\mathbb{E}_{p_t}[f(x)] \triangleq \mathbb{E}_{p_t}[\partial_t \log \tilde{p}_t(x)] \approx \frac{1}{K}\sum_{k=1}^K \partial_t \log \tilde{p}_t(x^{(k)})$, where $x^{(k)} \sim p_t$. Specifically, we define $g(x) = \nabla_x \cdot v_t(x;\theta) + v_t(x;\theta) \cdot \nabla_x \log p_t(x)$, from which we have $\gamma = \mathbb{E}_{p_t}[g(x)] = 0$. Using this, we construct a new estimator:

$$\check{\mu} = \frac{1}{K}\sum_{k=1}^K \partial_t \log \tilde{p}_t(x^{(k)}) + \beta^*(\nabla_x \cdot v_t(x^{(k)};\theta) + v_t(x^{(k)};\theta) \cdot \nabla_x \log p_t(x^{(k)})), \quad x^{(k)} \sim p_t. \tag{21}$$

Moreover, when θ is optimal, Equation (3) equals zero, implying $g(x) = -f(x) + c$, where c is a constant independent of the sample x. In this case, $\text{Corr}(f, g) = -1$, and $\check{\mu}$ becomes a zero-variance estimator. As an additional clarification, Stein's identity from Theorem 1 is also employed as a control variate in Liu et al. (2017), where it is utilized to optimise the policy in reinforcement learning.

Appendix C. Training and Sampling Algorithms

The training and sampling algorithms are detailed in Algorithms 2 and 3, respectively. For clarity, Algorithm 2 illustrates a single training epoch. In particular, we parameterise the model with a single neural network $s_t(x, d; \theta)$ that takes the sample x, time step t, and shortcut distance d as input to anticipate the shortcut direction. This design enables NFS^2 to model in continuous time, unlike the baseline LFIS (Tian et al., 2024), which trains separate neural networks for each time step — a memory-intensive and inefficient approach. To train the model, we define time spans $\{t_m\}_{m=0}^{M}$ that are evenly distributed over $[0, 1]$, satisfying $0 = t_0 < \cdots < t_M = 1$ and $2t_m = t_{m+1} + t_{m-1}, \forall m$. In each epoch, we uniformly sample time steps from the time spans $\tilde{t}_m \sim \mathcal{U}([t_{m-1}, t_m])$ and ensure that $\tilde{t}_0 = 0$.[3] Subsequently, Algorithm 1 is invoked to generate training samples, which resembles distribution q as defined in Equation (3). Notably, any q distribution can be used to generate training samples. The choice of the proposed velocity-driven SMC is motivated by two key reasons:

i) At the beginning of training, the generated samples are far from the mode, encouraging the model to focus on mode-covering. As training progresses, the generated samples become more accurate, gradually shifting toward mode-seeking, ultimately balancing exploration and exploitation for improved learning efficiency.

ii) Improved $\partial_t \log Z_t$ estimation efficiency. SMC returns the samples and importance weights for each time step simultaneously, streamlining the estimation of $\partial_t \log Z_t$.

After generating training samples, we compute the loss in Equation (5) and update the model using gradient descent, as outlined in steps 4–9 of Algorithm 2.

Appendix D. Experimental Details

D.1. Datasets

Gaussian Mixture Model (GMM-40). We use a 40 Gaussian mixture density in 2 dimensions as proposed by Midgley et al. (2023). This density consists of a mixture of 40 evenly weighted Gaussians with identical covariances

$$\Sigma = \begin{bmatrix} 40 & 0 \\ 0 & 40 \end{bmatrix}$$

and μ_i are uniformly distributed over the $[-40, 40]$ box, i.e., $\mu_i \sim U(-40, 40)^2$.

$$p(x) = \frac{1}{40} \sum_{i=1}^{40} \mathcal{N}(x; \mu_i, \Sigma)$$

Many Well 32 (MW-32). We use a 32-dimensional Many Well density, as proposed by Midgley et al. (2023). This density consists of a mixture of $n_{\text{wells}} = 16$ independent double-well potentials:

$$E(x) = \sum_{i=1}^{n_{\text{wells}}} E_{\text{DW}}(x_i)$$

3. More advanced schedule beyond uniform sampling remain important future works.

where each x_i corresponds to a pair of variables in a 2-dimensional space. The unnormalized log density for a single 2D Double Well is:

$$\log p_{\text{DW}}(x_1, x_2) = -x_1^4 + 6x_1^2 + \frac{1}{2}x_1 - \frac{1}{2}x_2^2$$

Here, the wells are symmetrically distributed across a grid in the 32-dimensional space, where each pair of dimensions corresponds to a well, and μ_i is uniformly distributed over the space. The total log probability is proportional to the sum of energies from all wells: $\log p(x) \propto E(x) = \sum_{i=1}^{n_{\text{wells}}} E_{\text{DW}}(x_i)$.

Double Well 4. The energy function for the DW-4 dataset was introduced in Köhler et al. (2020) and corresponds to a system of 4 particles in a 2-dimensional space. The system is governed by a double-well potential based on the pairwise distances of the particles. For a system of 4 particles, $x = \{x_1, \ldots, x_4\}$, the energy is given by:

$$E(x) = \frac{1}{2\tau} \sum_{i,j} \left[a(d_{ij} - d_0) + b(d_{ij} - d_0)^2 + c(d_{ij} - d_0)^4 \right]$$

where $d_{ij} = \|x_i - x_j\|_2$ is the Euclidean distance between particles i and j. Following previous work, we set $a = 0$, $b = -4$, $c = 0.9$, and the temperature parameter $\tau = 1$. To evaluate the efficacy of our samples, we use a validation and test set from the MCMC samples in Klein et al. (2024) as the ground truth samples following the practice of previous works (Sadegh et al., 2024).

D.2. Metrics

We evaluate the methods using the Wasserstein-2 ($\mathcal{W}_2$) distance and the Total Variation (TV), both computed with 1,000 ground truth and generated samples. To compute TV, the support is divided into 200 bins along each dimension, and the empirical distribution over 1,000 samples is used. For GMM-40, we report the metrics$\mathcal{W}_2$ on energy space $\mathcal{E}$ and TV on the data space $\mathcal{X}$. For MW-32, we find that $\mathcal{E} - \mathcal{W}_2$ is unstable and thus report $\mathcal{E}$-TV instead. Given the 32-dimensional nature of MW-32, computing TV is impractical; therefore, we report the $\mathcal{W}_2$ metric on the data space rather than TV. For the n-body system DW-4, we do not report any metrics in the data space due to its equivariance. Instead, we assess performance using metrics in the energy space ($\mathcal{E}$-$\mathcal{W}_2$ and $\mathcal{E}$-TV) and the interatomic coordinates $\mathcal{D}$ ($\mathcal{D}$-TV) to account for invariance.

D.3. Training Details

Gaussian Mixture Model (GMM-40). We evaluate our method on a 40-mode Gaussian mixture in $\mathbb{R}^2$ to test multi-modal exploration. The velocity field is parameterized by a 4-layer MLP (128-dimensional hidden layers, Layer Norm, and GeLU activations) trained using velocity-guided sequential Monte Carlo with Hamiltonian kernels (3 HMC steps, 5 leapfrog steps, step size $\eta = 0.1$). Initial particles are sampled from $\mathcal{N}(\mathbf{0}, 25\mathbf{I})$, optimized via AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$) with learning rate 4×10^{-4}, weight decay 10^{-4}, and gradient clipping at ℓ_2-norm 1.0. Training uses 128-particle batches for 10^4 epochs (500 steps/epoch) with early stopping, converging significantly before the epoch limit.

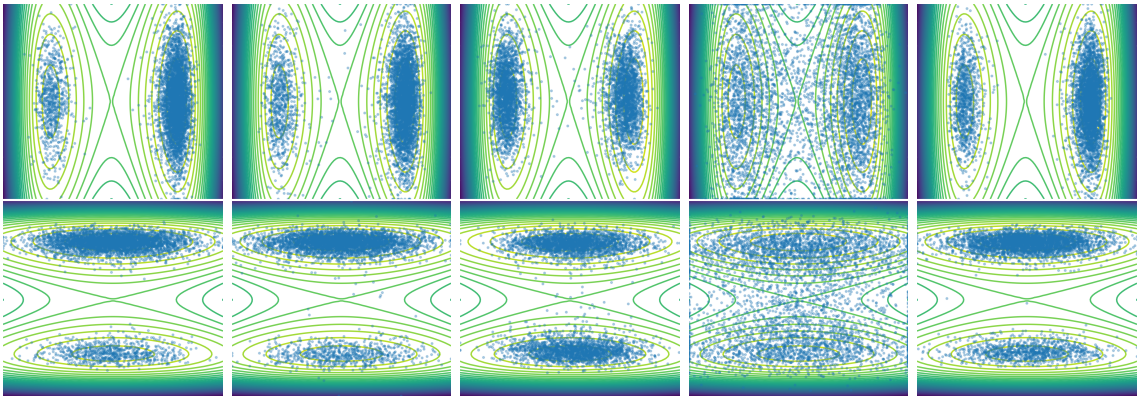


Figure 6: Samples on MW-32. First row: 2D marginal samples from the 1st and 4th dimensions; Second row: 2D marginal samples from the 2rd and 3rd dimensions.

Many Well 32 (MW-32). We assess scalability in high dimensions using a 2^{32}-mode Many Well potential on $\mathbb{R}^{32}$, exhibiting exponential mode growth with dimension. The velocity field employs a 4-layer MLP (128-dimensional hidden layers, Layer Norm, and GeLU activations) trained via velocity-guided SMC with enhanced Hamiltonian kernels (6 HMC steps, 10 leapfrog steps, step size $\eta = 0.1$). Initialized from $\mathcal{N}(\mathbf{0}, 2\mathbf{I})$, optimization uses AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$) with learning rate $1e^{-3}$, weight decay 10^{-4}, and ℓ_2-gradient clipping at 1.0. Training maintains 128-particle batches across 10^4 epochs (500 steps/epoch) with early stopping.

Double Well 4 (DW-4). We assess performance in particle-like system using a DW-4 potential on euclidean space. The velocity field employs a 4-layer MLP (512-dimensional hidden layers, Layer Norm, and GeLU activations) trained via velocity-guided SMC with enhanced Hamiltonian kernels (10 HMC steps, 10 leapfrog steps, step size $\eta = 0.01$). Initialized from $\mathcal{N}(\mathbf{0}, 2\mathbf{I})$, optimization uses AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$) with learning rate $4e^{-3}$, weight decay 10^{-4}, and ℓ_2-gradient clipping at 1.0. Training maintains 128-particle batches across 10^4 epochs (500 steps/epoch) with early stopping.

Appendix E. Additional Experimental Results

E.1. Visualisation of MW-32

This section presents additional visualizations of generated samples on MW-32. As shown in Figure 6, only FAB and NFS^2 accurately capture the modes. While iDEM locates the modes, it struggles to identify their correct weights. Additionally, LFIS, another flow-based sampler similar to NFS^2, produces noisy samples, high-

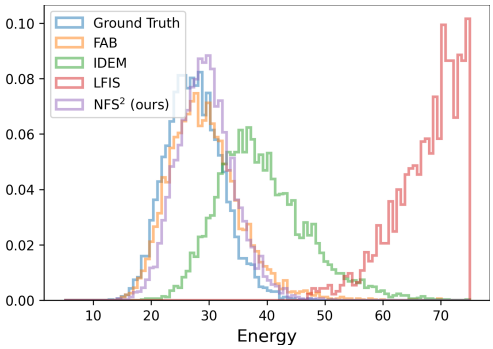


Figure 7: Histogram of sample energy on MW-32.

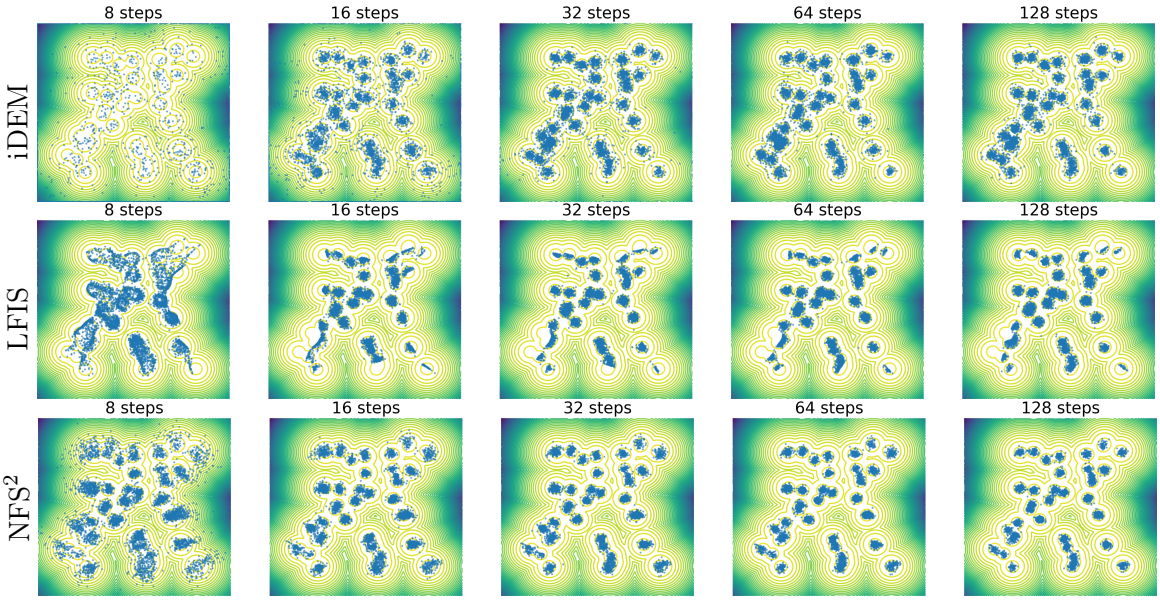


Figure 8: Illustration of the generated samples using different sampling steps on GMM-40.

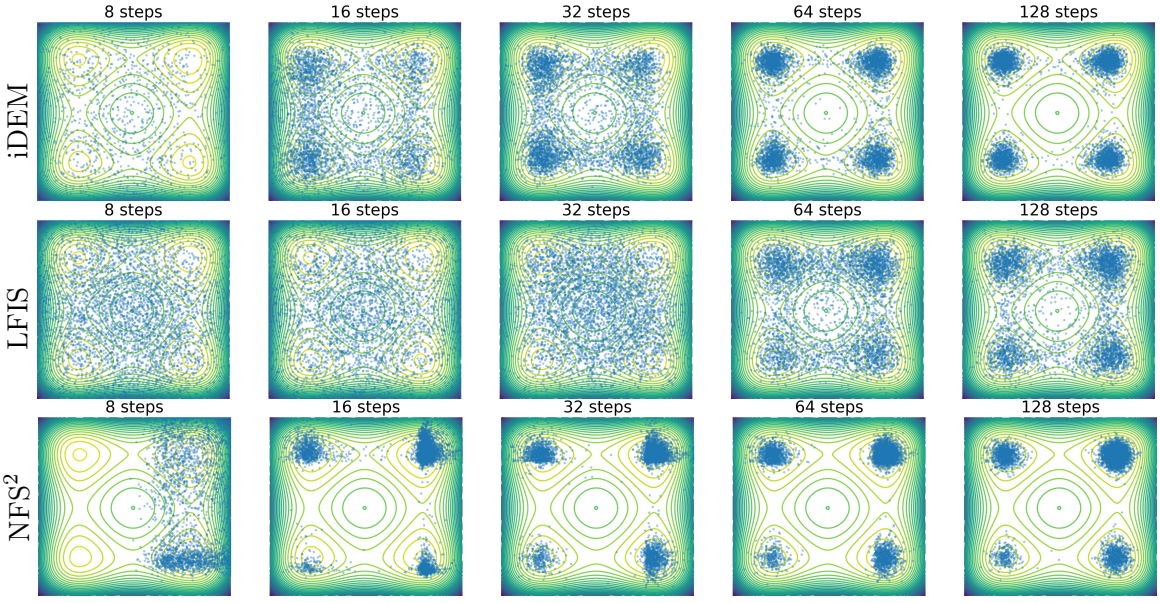


Figure 9: Illustration of the generated samples using different sampling steps on MW-32.

lighting the high variance issue associated with importance sampling. We further illustrate the histogram of sample energy on MW-32, where we draw the empirical energy distribution using 5,000 samples. It shows that NFS^2 achieves competitive performance with FAB, and notably outperforms iDEM and LFIS.

E.2. Comparisons with Different Sampling Steps

One key advantage of NFS^2 is its ability to achieve high-quality results with fewer sampling steps. In this section, we compare NFS^2 to the SOTA diffusion-based sampler iDEM and the flow-based sampler LFIS, using varying numbers of sampling steps. As demonstrated in Figures 8 and 9, NFS^2 produces better samples compared to both iDEM and LFIS, when using fewer sampling steps.