Adaptive Destruction Processes for Diffusion Samplers

Timofei Gritsaev[1,2*] **Nikita Morozov**[2] **Kirill Tamogashev**[3] **Daniil Tiapkin**[4,5]
Sergey Samsonov[2] **Alexey Naumov**[2] **Dmitry Vetrov**[1] **Nikolay Malkin**[3]
[1]Constructor University [2]HSE University [3]University of Edinburgh
[4]CMAP, École Polytechnique [5]LMO, Université Paris–Saclay

Abstract

In this paper, we explore the challenges and benefits of a trainable destruction process in diffusion samplers. We allow for additional flexibility in the definition of the generative and destruction policies, enabling both transition kernels to be learned as unconstrained Gaussian densities. We show that, when the number of steps is limited, our approach results in faster convergence and improved sampling quality on various benchmarks, and investigate the design choices necessary to facilitate stable training. Finally, we show the scalability of our approach through experiments on GAN latent space sampling for conditional image generation.

1 Introduction

Probabilistic inference is concerned with sampling from a distribution defined by an unnormalised density. Contrary to the generative modelling scenario, the learner has access only to an energy function $\mathcal{E}(x) : \mathbb{R}^d \to \mathbb{R}$, *without* access to ground truth samples. The aim is to sample from

$$p_{\text{target}}(x) = e^{-\mathcal{E}(x)}/Z; \quad Z = \int_{\mathbb{R}^d} e^{-\mathcal{E}(x)}\,\mathrm{d}x \tag{1}$$

and estimate the (typically intractable) normalising constant Z. *Amortised* solutions to this problem train a generative model to approximately sample from p_{target}, offering a more scalable alternative to Monte Carlo or MCMC algorithms [48, 50, 25]. The success of diffusion models as distribution approximators [69, 24, 70] motivate the application of diffusion-based techniques to amortised sampling. Unlike typical diffusion models, these *diffusion samplers* do not assume access to samples from the target density and are suitable for the sampling problem (1). There are many algorithms for training diffusion samplers (see §2.1 and Appendix A), many originating in the theory of stochastic control [*e.g.*, 88, 77] or in discrete-time reinforcement learning [37].

Most stochastic control objectives assume an underlying continuous-time dynamics, necessitating constraints on the generative and destruction processes: for example, their diffusion coefficients must coincide (see Appendix B.1). Based on a more flexible discrete-time formulation of diffusion samplers, we train both the generation and destruction processes. We find that faster and more accurate samplers are obtained by learning the kernels of both processes as Gaussian distributions with trainable mean and variance. We study choices of model architectures, training objectives, and techniques needed for stable training. Finally, we show that our results scale to higher-dimensional problems in experiments on text-conditional sampling in the latent space of a pretrained StyleGAN3 [31].

2 Methodology

2.1 Setting and background

To solve the problem (1), we simulate the process given by a (forward or generative) Itô SDE:

$$\mathrm{d}\overrightarrow{X_t} = \overrightarrow{f_\theta}(\overrightarrow{X_t}, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\overrightarrow{W_t}, \quad \overrightarrow{X_0} \sim p_0, \quad t \in [0, 1]\,, \tag{2}$$

*Correspondence to tgritsaev@constructor.university.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: 'Frontiers in Probabilistic Inference: Sampling Meets Learning' .

where p_0 is a tractable distribution (*e.g.*, Gaussian or Dirac). The drift function $\overrightarrow{f_\theta}$ is a neural network with parameters θ, and we want to find θ such that $\overrightarrow{X_1} \sim p_{\text{target}}$, *i.e.*, the terminal samples $\overrightarrow{X_1}$ are distributed as p_{target}. There may be many SDEs of the form in (2) stochastically transporting p_0 to p_{target}. To set a learning target for θ, we consider a backward (or destructive) SDE:

$$\mathrm{d}\overleftarrow{X_t} = \overleftarrow{f_\varphi}(\overleftarrow{X_t}, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\overleftarrow{W_t}, \quad \overleftarrow{X_1} \sim p_{\text{target}}. \tag{3}$$

Given a fixed process (3), there is a unique forward SDE of the form (2) that defines the same process, *i.e.*, the processes $\overrightarrow{X_t}$ and $\overleftarrow{X_t}$ coincide. In particular, the marginal distributions at time 1 would then coincide, so $\overrightarrow{X_1} \sim p_{\text{target}}$. Thus, training objectives aim to enforce *time reversal* – to minimise some divergence between the generation and destruction processes with respect to their parameters.

In order to learn the time-discrete variant of (2) we use a T-step Euler-Maruyama scheme to obtain a Markov chain: $X_0 \to X_{\Delta t} \to \ldots \to X_1$; $\Delta t = 1/T$.[2] The discrete dynamics can be written as:

$$X_{t+\Delta t} = X_t + f_\theta(X_t, t)\,\Delta t + g(t)\sqrt{\Delta t}\,\xi_t, \quad X_0 \sim p_0(X_0),\ \xi_t \sim \mathcal{N}(0, I_d) \tag{4}$$

with the conditional probability density for each timestep being

$$\overrightarrow{p_\theta}(X_{t+\Delta t} \mid X_t) = \mathcal{N}(X_{t+\Delta t}; X_t + f_\theta(X_t, t)\,\Delta t,\ g(t)^2 \Delta t\, I_d)\,. \tag{5}$$

The forward and reverse transition kernels $\overrightarrow{p_\theta}$ and $\overleftarrow{p_\varphi}$ induce joint distributions over $\mathbf{X} = (X_0, X_{\Delta t}, \ldots, X_1)$, corresponding to time-discrete variants of (2) and (3)[3], which have the form

$$\overrightarrow{p_\theta}(\mathbf{X}) = p_0(X_0) \prod_{i=1}^{T} \overrightarrow{p_\theta}(X_{i\Delta t} | X_{(i-1)\Delta t})\,,\ \overleftarrow{p_\varphi}(\mathbf{X}) = p_{\text{target}}(X_1) \prod_{i=1}^{T} \overleftarrow{p_\varphi}(X_{(i-1)\Delta t} | X_{i\Delta t})\,. \tag{6}$$

If p_0 is Dirac the transition $\overleftarrow{p_\varphi}(X_0 \mid X_{\Delta t})$ is understood to be Dirac and to have density 1 when it appears in objectives. At convergence we expect that approximately

$$p_0(X_0)\overrightarrow{p_\theta}(X_{\Delta t,\ldots,1} \mid X_0) = p_{\text{target}}(X_1)\overleftarrow{p_\varphi}(X_{0,\ldots,(T-1)\Delta t} \mid X_1)\,, \tag{7}$$

or, for brevity, $p_0\overrightarrow{p_\theta} = p_{\text{target}}\overleftarrow{p_\varphi}$. The majority of past work (see Appendix A) focuses on learning the generation process given a fixed destruction process, although some [59] derive objectives for optimising both processes. We propose to exploit the flexibility of time-discrete formulation of diffusion samplers to parametrise both drift and variance of generation and destruction processes using neural networks. We detail our approach in §2.2 and 2.3 and show experimental findings in §3.

2.2 Relaxing constraints on transitions

We first propose to relax the constraint that the generation and destruction processes arise from a pair of SDEs (2) and (3) with fixed time-dependent variance. To achieve this, we allow both generatitunon and destruction processes to be Gaussian with arbitrary means and variances, while maintaining the requirement that the destruction process leads to p_0 at the last step. This is done by learning the generation and destruction means and variances as corrections to those arising in time-discrete variants of (2) and (3). By introducing corrections we get the following generative transition kernel:

$$\overrightarrow{p_\theta}(X_{t+\Delta t} \mid X_t) = \mathcal{N}(X_{t+\Delta t}; X_t + f_\theta(X_t, t)\,\Delta t,\ \mathrm{diag}(\gamma_\theta(X_t, t))\sigma^2 \Delta t\, I_d)\,, \tag{8}$$

and the following destruction kernel:

$$\overleftarrow{p_\varphi}(X_t \mid X_{t+\Delta t}) = \mathcal{N}\left(X_t; \mathrm{diag}(\alpha_\varphi(X_t, t))\,\frac{t}{t+\Delta t} X_{t+\Delta t},\ \mathrm{diag}(\beta_\varphi(X_t, t))\,\frac{t}{t+\Delta t}\sigma^2 \Delta t\, I_d\right) \tag{9}$$

where $\alpha_\varphi(X_t, t)$, $\beta_\varphi(X_t, t)$, $\gamma_\theta(X_t, t)$ are neural networks with d-dimensional outputs. Details on the functional form of these functions are given in Appendix B.3. In theory, the transitions could be more general than Gaussian, as long as their density is tractable to sample and differentiate.

The ability to train the destruction process increases the degrees of freedom in the feasible space of solutions $(p_0\overrightarrow{p_\theta}, p_{\text{target}}\overleftarrow{p_\varphi})$ when minimising the discrepancy in the time-reversal condition (7). Meanwhile, training the variance of the generation process corrects for the poor approximation to the reverse of a fixed destruction process by Gaussian transition with the same variances when the time discretisation is coarse.

[2] For notational clarity we assume a uniform discretisation, but all derivations hold for variable time steps.

[3] This can use the reverse Euler-Maruyama scheme, or (as in [88] and other works, including ours) the reversal of a forward-time scheme, which do not coincide, but converge as $\Delta t \to 0$; see, *e.g.*, [7, Prop. 3.5].

2.3 Training objectives for generation and destruction processes

We consider a number of training objectives from the previous literature with both fixed and learnable destruction processes. Learning discrete Markov process requires optimising a divergence between two distributions $\mathbb{D}(p_0\overrightarrow{p_\theta}\|p_{\text{target}}\overleftarrow{p_\varphi})$ that enforces equality in (7). The seminal works [88, 77] proposed to optimize **reverse KL** divergence $\mathbb{D}_{\text{KL}}$ through reparameterisation trick. Although this scheme yields an unbiased estimator, it requires backpropagating though the simulation of the generation process. Alternatively, one can use second-moment divergences where the expectation is taken with respect to an arbitrary proposal distribution $\tilde{p}$. The losses that fall under this category are **trajectory balance** [**TB**; 41]) and **VarGrad** [59]. They alleviate the need for backpropagating though the simulation of the generation process, as well as allow for using off-policy exploration techniques [64]. TB and VarGrad inherently allow for end-to-end learning of generation process and destruction process, making them a straightforward choice for our purposes. In addition, reverse KL can also be used for optimization of $\overleftarrow{p_\varphi}$, resulting in an approach that is called **trajectory likelihood maximisation** [**TLM**; 21] in the more general GFlowNet setting.

Table 1 summarises the possible combinations of objectives for destruction and generation policies. In §3 we numerically study the behaviour of the aforementioned objectives. The detailed descriptions of training objectives are presented in Appendix B.2.

Table 1: Summary of objectives for learning generation and destruction processes.

	$\overrightarrow{p_\theta}$ (generation)	
$\overleftarrow{p_\varphi}$ (destruction) ↓	2^{nd} moment	Reverse KL
Fixed	[59, 64]	PIS [88], DDS [77]
2^{nd} moment	$\text{TB}_{\theta,\varphi}$	PIS_θ + VarGrad_φ
Rev. KL	TB_θ + TLM_φ	PIS_θ + TLM_φ

2.4 Techniques for stable joint optimisation

We found that simultaneously training generation and destruction processes requires a number of careful design choices in training. Using a **shared MLP backbone** for predicting mean and log-variance of generation and destruction processes, as well us utilizing **separate optimizers** to account for different scales of gradients when different training objectives are used for the generation and destruction process, facilitates faster convergence. Since the destruction process sets a learning target for the generation process, making it learnable causes this target to become nonstationary, leading to instabilities in optimisation. Thus, **tuning relative learning rates** is critical for stable training [21]. In addition, we use **target networks** for φ when computing the loss gradient for θ and vice versa, which is a well-know RL [44] technique that further facilitates stability. We also use the existing **exploration techniques** proposed by [37, 64], namely, increasing the variance of the generation process for sampling trajectories for training, as well as local search based on Langevin dynamics. Finally, we employ **prioritised experience replay** [PER; 63] with prioritisation by the loss, which was also utilised for GFlowNet training by [73]. Detailed descriptions are presented in Appendix C.1, and a numerical study is carried out in Appendix C.2.

3 Experiments

3.1 Synthetic tasks

Energies and metrics. We consider several target densities that are commonly used in the diffusion samplers literature, as well as more challenging variants of these distributions (see Appendix E.1 for details): **Gaussian mixture models (GMM)**, **Funnel** [49], and **ManyWell** [53]. Following the literature, we use the ELBO and EUBO [10] metrics to evaluate trained samplers. We also compute the 2-Wasserstein distance between generated and ground truth samples.

Algorithms compared. We use two objectives for optimisation of the generation process: trajectory balance (TB_θ, [41]) and reverse KL (equivalent to PIS [88] and so denoted PIS_θ). With each generation process objective, we compare four settings and vary the number of discretisation steps T:

- Fixed generation variances, fixed destruction process (as in most prior work);
- Learnable generation variances, fixed destruction process;
- Learnable generation variances, destruction process learned using reverse KL (TLM_φ) or TB (TB_φ). (When the generation process is learned using reverse KL, we replace TB_φ by VarGrad_φ, since the log-normalising constant is not learned.)

All training details can be found in Appendix E.3.

Results. Representative results are shown in Figs. 1 and 3 and full tables are in Appendix F. We summarise the main conclusions below. Extended discussion can be found in Appendix D.1.

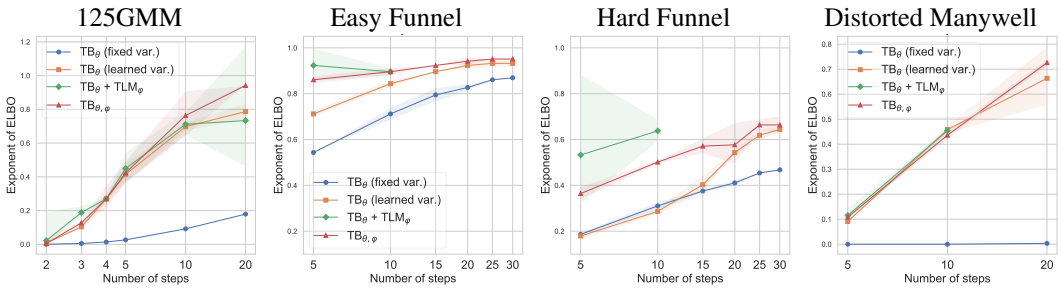

Figure 1: We compare performance on **125GMM**, **Easy Funnel**, **Hard Funnel**, **Distorted Manywell** of TB$_\theta$ (fixed var.), TB$_\theta$ (learned var.), TB$_\theta$ + TLM$_\varphi$, TB$_{\theta,\varphi}$ (other objectives in Fig. 3, Appendix F). Results are compared using ELBO. Mean and std over 3 seeds, collapsed runs excluded.

Learned generation variances improve sampling. Learnable variance of the generation process significantly improves the performance of the sampler, especially when the number of generation steps is small (Fig. 1) and in environments with high distortions (Fig. 4).

Off-policy losses are superior to differentiable simulation. The off-policy TB loss with exploratory behaviour policy is on par with or better than the PIS loss for learning the generation process in all cases (Appendix F), showing that findings in past work, such as [59, 64, 34], continue to hold when generation variances and the destruction process are learned.

Learning the destruction process is beneficial. Learning the destruction process yields an improvement over models with fixed destruction process and learned generation variance on all tasks.

TB is preferable to TLM for learning destruction. Training the destruction process with the TLM loss is typically superior to TB for small T but unstable for large T (see Appendix F), which partially confirms the results of [21] in discrete-space cases.

3.2 Scalability: Sampling in GAN latent space for conditional image generation

To validate our method in a high-dimensional setting, we sampled latent vectors for a pretrained StyleGAN3 generator to produce face images aligned with a text prompt as in [81]. The target distribution is defined by an energy function combining a standard prior $\mathcal{N}(0, I)$ with the ImageReward score, which measures text-image alignment. We train 5-step diffusion samplers to draw from this distribution for 7 different text prompts, comparing our proposed approach with learnable variances (TB$_{\theta,\varphi}$) against a baseline with fixed variances [81] (TB$_\theta$).

We evaluate performance using ELBO, average ImageReward score, and diversity. Our method shows improved ELBO and ImageReward scores on 5 out of 7 prompts, similar performance on one, and a slight degradation on another. A representative example of the improvement is shown in Fig. 2. Average metrics are reported in Table 2. Extended discussion is in Appendix D.2.

Table 2: FFHQ text-conditional sampling results. All models use $T = 5$ discretisation steps.

	ELBO (↑)	$\mathbb{E}[\log r(\mathbf{x}, \mathbf{y})]$ (↑)	CLIP Diversity (↑)
Prior	−117.0	−1.17	0.36
TB$_\theta$ (fixed var.)	98.8	1.42	**0.24**
TB$_{\theta,\varphi}$	**104.5**	**1.49**	**0.24**

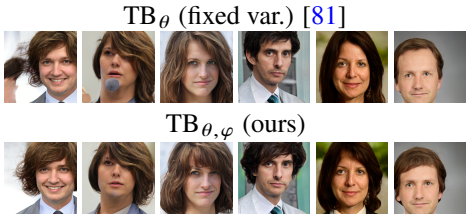

Figure 2: Decoded latents sampled with the same random seeds from outsourced diffusion samplers trained with a StyleGAN3 prior and ImageReward with prompt 'A person with medium length hair'.

4 Conclusion

In this paper we present the benefits of using learnable variance and learnable destruction process in diffusion samplers. We empirically find that these modifications help to more accurately model complex energy landscapes, especially with few sampling steps. We also contribute to the understanding of training diffusion samplers by studying techniques that improve their stability and convergence speed. We hope that these results inspire the community to scale our findings to other distributions and domains. Another interesting direction for future work would be to rigorously study the optimal parametrisations of the generation and destruction processes – including non-Gaussian transitions – and the theoretical limits of sampling with discrete-time learned diffusions.

Acknowledgment

The work of N.Morozov, A.Naumov, and S.Samsonov was supported by the grant for research centers in the field of AI provided by the Ministry of Economic Development of the Russian Federation in accordance with the agreement 000000C313925P4E0002 and the agreement with HSE University № 139-15-2025-009.

References

[1] Akhound-Sadegh, T., Rector-Brooks, J., Bose, A. J., Mittal, S., Lemos, P., Liu, C.-H., Sendera, M., Ravanbakhsh, S., Gidel, G., Bengio, Y., Malkin, N., and Tong, A. (2024). Iterated denoising energy matching for sampling from Boltzmann densities. *International Conference on Machine Learning (ICML)*.

[2] Atanackovic, L. and Bengio, E. (2025). Investigating generalization behaviours of generative flow networks. *Transactions on Machine Learning Research*.

[3] Bartosh, G., Vetrov, D., and Naesseth, C. A. (2024a). Neural diffusion models. *International Conference on Machine Learning (ICML)*.

[4] Bartosh, G., Vetrov, D. P., and Andersson Naesseth, C. (2024b). Neural flow diffusion models: Learnable forward process for improved diffusion modelling. *Neural Information Processing Systems (NeurIPS)*.

[5] Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. (2021). Flow network based generative models for non-iterative diverse candidate generation. *Neural Information Processing Systems (NeurIPS)*.

[6] Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. (2023). GFlowNet foundations. *Journal of Machine Learning Research*, 24(210):1–55.

[7] Berner, J., Richter, L., Sendera, M., Rector-Brooks, J., and Malkin, N. (2025). From discrete-time policies to continuous-time diffusion samplers: Asymptotic equivalences and faster training. *arXiv preprint arXiv:2501.06148*.

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

[9] Blessing, D., Berner, J., Richter, L., and Neumann, G. (2025). Underdamped diffusion bridges with applications to sampling. *International Conference on Learning Representations (ICLR)*.

[10] Blessing, D., Jia, X., Esslinger, J., Vargas, F., and Neumann, G. (2024). Beyond ELBOs: A large-scale evaluation of variational methods for sampling. *International Conference on Machine Learning (ICML)*.

[11] Bou, A., Bettini, M., Dittert, S., Kumar, V., Sodhani, S., Yang, X., De Fabritiis, G., and Moens, V. (2023). TorchRL: A data-driven decision-making library for PyTorch. *arXiv preprint arXiv:2306.00577*.

[12] Brunswic, L., Li, Y., Xu, Y., Feng, Y., Jui, S., and Ma, L. (2024). A theory of non-acyclic generative flow networks. *Association for the Advancement of Artificial Intelligence (AAAI)*, 38(10).

[13] Chen, T., Liu, G.-H., and Theodorou, E. A. (2022). Likelihood training of Schrödinger bridge using forward-backward SDEs theory. *International Conference on Learning Representations (ICLR)*.

[14] Cretu, M., Harris, C., Igashov, I., Schneuing, A., Segler, M., Correia, B., Roy, J., Bengio, E., and Lio, P. (2025). SynFlowNet: Design of diverse and novel molecules with synthesis constraints. *International Conference on Learning Representations (ICLR)*.

[15] De Bortoli, V., Thornton, J., Heng, J., and Doucet, A. (2021). Diffusion Schrödinger bridge with applications to score-based generative modeling. *Neural Information Processing Systems (NeurIPS)*.

[16] Deleu, T., Góis, A., Emezue, C., Rankawat, M., Lacoste-Julien, S., Bauer, S., and Bengio, Y. (2022). Bayesian structure learning with generative flow networks. *Uncertainty in Artificial Intelligence (UAI)*.

[17] Deleu, T., Nishikawa-Toomey, M., Subramanian, J., Malkin, N., Charlin, L., and Bengio, Y. (2023). Joint Bayesian inference of graphical structure and parameters with a single generative flow network. *Neural Information Processing Systems (NeurIPS)*.

[18] Deleu, T., Nouri, P., Malkin, N., Precup, D., and Bengio, Y. (2024). Discrete probabilistic inference as control in multi-path environments. *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*, pages 997–1021.

[19] Geist, M., Scherrer, B., and Pietquin, O. (2019). A theory of regularized Markov decision processes. *International Conference on Machine Learning (ICML)*.

[20] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Neural Information Processing Systems (NeurIPS)*.

[21] Gritsaev, T., Morozov, N., Samsonov, S., and Tiapkin, D. (2025). Optimizing backward policies in GFlownets via trajectory likelihood maximization. *International Conference on Learning Representations (ICLR)*.

[22] Havens, A., Miller, B. K., Yan, B., Domingo-Enrich, C., Sriram, A., Wood, B., Levine, D., Hu, B., Amos, B., Karrer, B., Fu, X., Liu, G.-H., and Chen, R. T. Q. (2025). Adjoint sampling: Highly scalable diffusion samplers via adjoint matching. *arXiv preprint arXiv:2504.11713*.

[23] Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus).

[24] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Neural Information Processing Systems (NeurIPS)*, 33:6840–6851.

[25] Hoffman, M. D., Gelman, A., et al. (2014). The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623.

[26] Hu, E. J., Jain, M., Elmoznino, E., Kaddar, Y., Lajoie, G., Bengio, Y., and Malkin, N. (2024). Amortizing intractable inference in large language models. *International Conference on Learning Representations (ICLR)*.

[27] Hu, E. J., Malkin, N., Jain, M., Everett, K., Graikos, A., and Bengio, Y. (2023). GFlowNet-EM for learning compositional latent variable models. *International Conference on Machine Learning (ICML)*.

[28] Jain, M., Bengio, E., Hernandez-Garcia, A., Rector-Brooks, J., Dossou, B. F., Ekbote, C. A., Fu, J., Zhang, T., Kilgour, M., Zhang, D., Simine, L., Das, P., and Bengio, Y. (2022). Biological sequence design with GFlowNets. *International Conference on Machine Learning (ICML)*.

[29] Jang, H., Jang, Y., Kim, M., Park, J., and Ahn, S. (2024a). Pessimistic backward policy for GFlowNets. *Neural Information Processing Systems (NeurIPS)*.

[30] Jang, H., Kim, M., and Ahn, S. (2024b). Learning energy decompositions for partial inference in GFlownets. *International Conference on Learning Representations (ICLR)*.

[31] Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., and Aila, T. (2021). Alias-free generative adversarial networks. *Neural Information Processing Systems (NeurIPS)*.

[32] Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. *Computer Vision and Pattern Recognition (CVPR)*.

[33] Kim, M., Choi, S., Kim, H., Son, J., Park, J., and Bengio, Y. (2025a). Ant colony sampling with GFlowNets for combinatorial optimization. *Artificial Intelligence and Statistics (AISTATS)*.

[34] Kim, M., Choi, S., Yun, T., Bengio, E., Feng, L., Rector-Brooks, J., Ahn, S., Park, J., Malkin, N., and Bengio, Y. (2025b). Adaptive teachers for amortized samplers. *International Conference on Learning Representations (ICLR)*.

[35] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*.

[36] Kong, L., Cui, J., Sun, H., Zhuang, Y., Prakash, B. A., and Zhang, C. (2023). Autoregressive diffusion model for graph generation. *International Conference on Machine Learning (ICML)*.

[37] Lahlou, S., Deleu, T., Lemos, P., Zhang, D., Volokhova, A., Hernández-Garcıa, A., Ezzine, L. N., Bengio, Y., and Malkin, N. (2023). A theory of continuous generative flow networks. *International Conference on Machine Learning (ICML)*.

[38] Lee, S., Kim, M., Cherif, L., Dobre, D., Lee, J., Hwang, S. J., Kawaguchi, K., Gidel, G., Bengio, Y., Malkin, N., and Jain, M. (2025). Learning diverse attacks on large language models for robust red-teaming and safety tuning. *International Conference on Learning Representations (ICLR)*.

[39] Madan, K., Lamb, A., Bengio, E., Berseth, G., and Bengio, Y. (2025). Towards improving exploration through sibling augmented GFlowNets. *International Conference on Learning Representations (ICLR)*.

[40] Madan, K., Rector-Brooks, J., Korablyov, M., Bengio, E., Jain, M., Nica, A. C., Bosc, T., Bengio, Y., and Malkin, N. (2023). Learning GFlowNets from partial episodes for improved convergence and stability. *International Conference on Machine Learning (ICML)*.

[41] Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. (2022). Trajectory balance: Improved credit assignment in GFlowNets. *Neural Information Processing Systems (NeurIPS)*.

[42] Malkin, N., Lahlou, S., Deleu, T., Ji, X., Hu, E. J., Everett, K. E., Zhang, D., and Bengio, Y. (2023). GFlowNets and variational inference. *International Conference on Learning Representations (ICLR)*.

[43] Midgley, L. I., Stimper, V., Simm, G. N., Schölkopf, B., and Hernández-Lobato, J. M. (2023). Flow annealed importance sampling bootstrap. *International Conference on Learning Representations (ICLR)*.

[44] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

[45] Mohammadpour, S., Bengio, E., Frejinger, E., and Bacon, P.-L. (2024). Maximum entropy GFlowNets with soft Q-learning. *Artificial Intelligence and Statistics (AISTATS)*.

[46] Morozov, N., Maksimov, I., Tiapkin, D., and Samsonov, S. (2025). Revisiting non-acyclic GFlowNets in discrete environments. *International Conference on Machine Learning (ICML)*.

[47] Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. (2017). Bridging the gap between value and policy based reinforcement learning. *Neural Information Processing Systems (NeurIPS)*.

[48] Neal, R. M. (1998). Annealed importance sampling. *arXiv preprint arXiv:physics/9803008*.

[49] Neal, R. M. (2003). Slice sampling. *The annals of statistics*, 31(3):705–767.

[50] Neal, R. M. et al. (2011). MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2.

[51] Neu, G., Jonsson, A., and Gómez, V. (2017). A unified view of entropy-regularized Markov decision processes. *arXiv preprint arXiv:1705.07798*.

[52] Nielsen, B. M., Christensen, A., Dittadi, A., and Winther, O. (2024). DiffEnc: Variational diffusion with a learned encoder. *International Conference on Learning Representations (ICLR)*.

[53] Noé, F., Olsson, S., Köhler, J., and Wu, H. (2019). Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147.

[54] Pan, L., Malkin, N., Zhang, D., and Bengio, Y. (2023). Better training of GFlowNets with local credit and incomplete trajectories. *International Conference on Machine Learning (ICML)*.

[55] Phillips, D. and Cipcigan, F. (2024). MetaGFN: Exploring distant modes with adapted metadynamics for continuous GFlowNets. *arXiv preprint arXiv:2408.15905*.

[56] Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[57] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. *International Conference on Machine Learning (ICML)*.

[58] Rector-Brooks, J., Madan, K., Jain, M., Korablyov, M., Liu, C.-H., Chandar, S., Malkin, N., and Bengio, Y. (2023). Thompson sampling for improved exploration in GFlowNets. *arXiv preprint arXiv:2306.17693*.

[59] Richter, L. and Berner, J. (2024). Improved sampling via learned diffusions. *International Conference on Learning Representations (ICLR)*.

[60] Richter, L., Boustati, A., Nüsken, N., Ruiz, F., and Akyildiz, O. D. (2020). VarGrad: a low-variance gradient estimator for variational inference. *Neural Information Processing Systems (NeurIPS)*.

[61] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241.

[62] Sahoo, S., Gokaslan, A., De Sa, C. M., and Kuleshov, V. (2024). Diffusion models with learned adaptive noise. *Neural Information Processing Systems (NeurIPS)*.

[63] Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). Prioritized experience replay. *International Conference on Learning Representations (ICLR)*.

[64] Sendera, M., Kim, M., Mittal, S., Lemos, P., Scimeca, L., Rector-Brooks, J., Adam, A., Bengio, Y., and Malkin, N. (2024). Improved off-policy training of diffusion samplers. *Neural Information Processing Systems (NeurIPS)*.

[65] Shen, T., Seo, S., Lee, G., Pandey, M., Smith, J. R., Cherkasov, A., Kim, W. Y., and Ester, M. (2024). TacoGFN: Target-conditioned GFlowNet for structure-based drug design. *Transactions on Machine Learning Research*.

[66] Shi, Y., Bortoli, V. D., Campbell, A., and Doucet, A. (2023). Diffusion Schrödinger bridge matching. *Neural Information Processing Systems (NeurIPS)*.

[67] Silva, T., Alves, R. B., de Souza da Silva, E., Souza, A. H., Garg, V., Kaski, S., and Mesquita, D. (2025). When do GFlowNets learn the right distribution? *International Conference on Learning Representations (ICLR)*.

[68] Silva, T., de Souza, D. A., and Mesquita, D. (2024). Streaming Bayes GFlowNets. *Neural Information Processing Systems (NeurIPS)*.

[69] Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *International Conference on Machine Learning (ICML)*.

[70] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations (ICLR)*.

[71] Song, Z., Yang, C., Wang, C., An, B., and Li, S. (2024). Latent logic tree extraction for event sequence explanation from LLMs. *International Conference on Machine Learning (ICML)*.

[72] Stromme, A. (2023). Sampling from a Schrödinger bridge. *Artificial Intelligence and Statistics (AISTATS)*.

[73] Tiapkin, D., Morozov, N., Naumov, A., and Vetrov, D. P. (2024). Generative flow networks as entropy-regularized RL. *Artificial Intelligence and Statistics (AISTATS)*.

[74] Tong, A., Malkin, N., Fatras, K., Atanackovic, L., Zhang, Y., Huguet, G., Wolf, G., and Bengio, Y. (2024). Simulation-free schrödinger bridges via score and flow matching. *Artificial Intelligence and Statistics (AISTATS)*.

[75] Uehara, M., Zhao, Y., Biancalani, T., and Levine, S. (2024). Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review. *arXiv preprint arXiv:2407.13734*.

[76] van Krieken, E., Thanapalasingam, T., Tomczak, J., Van Harmelen, F., and Ten Teije, A. (2023). A-NeSI: A scalable approximate method for probabilistic neurosymbolic inference. *Neural Information Processing Systems (NeurIPS)*.

[77] Vargas, F., Grathwohl, W., and Doucet, A. (2023). Denoising diffusion samplers. *International Conference on Learning Representations (ICLR)*.

[78] Vargas, F., Ovsianas, A., Fernandes, D., Girolami, M., Lawrence, N. D., and Nüsken, N. (2022). Bayesian learning via neural schrödinger–föllmer flows. *Statistics and Computing*, 33(1):3.

[79] Vargas, F., Padhy, S., Blessing, D., and Nüsken, N. (2024). Transport meets variational inference: Controlled monte carlo diffusions. *International Conference on Learning Representations (ICLR)*.

[80] Vargas, F., Thodoroff, P., Lamacraft, A., and Lawrence, N. (2021). Solving Schrödinger bridges via maximum likelihood. *Entropy*, 23(9):1134.

[81] Venkatraman, S., Hasan, M., Kim, M., Scimeca, L., Sendera, M., Bengio, Y., Berseth, G., and Malkin, N. (2025). Outsourced diffusion sampling: Efficient posterior inference in latent spaces of generative models. *International Conference on Machine Learning (ICML)*.

[82] Xu, J., Liu, X., Wu, Y., Tong, Y., Li, Q., Ding, M., Tang, J., and Dong, Y. (2023). Imagereward: Learning and evaluating human preferences for text-to-image generation. *Neural Information Processing Systems (NeurIPS)*.

[83] Younsi, A., Abubaker, A., Seddik, M. E. A., Hacid, H., and Lahlou, S. (2025). Accurate and diverse LLM mathematical reasoning via automated PRM-guided GFlowNets. *arXiv preprint arXiv:2504.19981*.

[84] Zhang, D., Chen, R. T. Q., Liu, C.-H., Courville, A., and Bengio, Y. (2024). Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. *International Conference on Learning Representations (ICLR)*.

[85] Zhang, D., Chen, R. T. Q., Malkin, N., and Bengio, Y. (2023a). Unifying generative models with GFlowNets and beyond. *arXiv preprint arXiv:2209.02606*.

[86] Zhang, D., Dai, H., Malkin, N., Courville, A., Bengio, Y., and Pan, L. (2023b). Let the flows tell: Solving graph combinatorial problems with GFlowNets. *Neural Infromation Processing Systems (NeurIPS)*.

[87] Zhang, D., Rainone, C., Peschl, M., and Bondesan, R. (2023c). Robust scheduling with GFlowNets. *International Conference on Learning Representations (ICLR)*.

[88] Zhang, Q. and Chen, Y. (2022). Path integral sampler: A stochastic control approach for sampling. *International Conference on Learning Representations (ICLR)*.

[89] Zhou, M. Y., Yan, Z., Layne, E., Malkin, N., Zhang, D., Jain, M., Blanchette, M., and Bengio, Y. (2024). PhyloGFN: Phylogenetic inference with generative flow networks. *International Conference on Learning Representations (ICLR)*.

[90] Zimmermann, H., Lindsten, F., van de Meent, J.-W., and Naesseth, C. A. (2023). A variational perspective on generative flow networks. *Transactions on Machine Learning Research*.

A Related work

Diffusion samplers. While diffusion models were first introduced with the goal of approximating a distribution from which samples are available by an iterative denoising process [69, 24, 70], *diffusion samplers* – on which the modern line of work started with [78, 88, 77] and continues to rapidly expand [79, 59, 1, 64, 9, 22, *inter alia*] – seek to amortise the cost of sampling from a given target density p_{target}, which can be queried for the energy and possibly its gradient, by training a generation process to approximate it. Algorithms such as PIS [88] and DDS [77] achieve this by minimising the KL divergence between destruction and generation processes. Other divergences have been considered for their better numerical properties, for example, [59] uses a second-moment divergence [60]. In [37, 85], diffusion samplers are understood as a generalised instance of GFlowNets [5], which are deep reinforcement learning algorithms that allow for using exploration techniques and off-policy training and do not require access to the gradients of p_{target} (although physics-informed parametrisations may use this gradient). Subsequent work [84, 64, 34] has applied training objectives from GFlowNet literature [41, 40, 54] to the diffusion sampling case and explored the benefits of the flexible off-policy training that GFlowNets allow [64, 55, 34]. Finally, [7] builds theoretical connections among various diffusion sampling objectives and their continuous-time limits.

GFlowNets. Generative flow networks [GFlowNets; 5, 6] have been introduced as a general framework for sampling from distributions by solving a sequential decision-making problem, training a stochastic policy to construct objects step by step. GFlowNets represent a synthesis of variational inference and reinforcement learning (RL) paradigms [42, 90, 73, 18]. Beyond their original application in biological structure discovery [*e.g.*, 28, 65, 14], GFlowNets have found uses in probabilistic inference over symbolic latent variables [16, 76, 27, 17, 89, 68], combinatorial optimisation [87, 86, 33], and reasoning or planning in language [26, 71, 38, 83]. Although initially defined for discrete spaces, the framework can be extended to continuous domains [37] and non-acyclic state spaces (*i.e.*, nonconstrutive actions) [12, 46]. GFlowNets approach the training of amortised samplers from a RL perspective and correspondingly require solving the RL challenges of exploration [58, 64, 55, 39, 34], credit assignment [41, 54, 30], and generalisation [67, 2]. Relevant to our work, design and training of the destruction process in *discrete* problems is a focus of [42, 45, 29, 21], where learning the destruction process is shown to improve convergence and mode discovery.

Learning the destruction process in diffusion. A recent line of work [3, 4, 62, 52] studies learning the destruction process (called the 'forward' process, clashing with the opposite convention in place for diffusion samplers) for diffusion models in image domains, instead of using a fixed one, such as VP or VE SDE [70]. This results in a broader class of models and is shown to improve log-likelihood and visual quality of the generated samples, as well as reducing the required number of training steps. GFlowNet objectives were applied to this problem in [37, 85]. Learning the destruction process for diffusion models was also studied in discrete domains, *e.g.*, [36] learns the order in which nodes are removed from a graph by the destruction process.

Our work is also adjacent to the **Schrödinger bridge (SB) problem**. In continuous time, the learning of destruction and generation processes that stochastically transport a source distribution p_0 to p_{target} is known as a *bridge problem*. The SB problem seeks the unique bridge that is closest to some reference process and can be solved by algorithms based on iterative proportional fitting in a time discretisation [IPF; 80, 15] or various other methods [13, 72, 66, 74]. Unlike diffusion samplers, these algorithms typically assume samples from p_{target} are available. Many of these algorithms, when typical approximations are made in training, at convergence yield bridges that are not necessarily the SB. In this work, we constrain the destruction process to be a bridge from p_{target} to p_0, meaning that in the continuous-time limit iterative proportional fitting would converge in a single step. However, the discrete-time TLM update for the destruction process (§2.3), which trains $\overleftarrow{p_\varphi}$ on trajectories sampled from $\overrightarrow{p_\theta}$, is in fact equivalent to an (unconverged) maximum-likelihood IPF step as in [80].

B More on diffusion samplers

B.1 On KL divergence between processes with different variances.

If two path space measures $\mathbb{P}_1$ and $\mathbb{P}_2$ defined by SDEs have different diffusion coefficients $g_1(t)$ and $g_2(t)$, they are not absolutely continuous with respect to each other, therefore, $\mathrm{KL}(\mathbb{P}_1 \,\|\, \mathbb{P}_2) = \infty$. To see this, notice that the quadratic variation of $\mathbb{P}_i$ on the time interval $[t, t']$ is almost surely

$\int_t^{t'} g_i(s)^2 \mathrm{d}s$. Therefore, if $\mathbb{P}_1 \ll \mathbb{P}_2$, then the $\int_t^{t'} g_1(s)^2 \mathrm{d}s = \int_t^{t'} g_2(s)^2 \mathrm{d}s$ for every $t < t'$, which implies $g_1 = g_2$ if both are continuous.

B.2 Training objectives.

Learning discrete Markov process requires optimising a divergence between two distributions $\mathbb{D}(p_0 \overrightarrow{p_\theta} \| p_{\text{target}} \overleftarrow{p_\varphi})$ that enforces equality in (7). One option can be a reverse KL divergence:

$$\begin{aligned}\mathbb{D}_{\text{KL}}(p_0 \overrightarrow{p_\theta} \| p_{\text{target}} \overleftarrow{p_\varphi}) &= \text{KL}\left(p_0(X_0)\overrightarrow{p_\theta}(X_{\Delta t,\dots,1} \mid X_0) \| p_{\text{target}}(X_1)\overleftarrow{p_\varphi}(X_{0,\dots,(T-1)\Delta t} \mid X_1)\right) \\ &= \mathbb{E}_{X_{0,\Delta t,\dots,1} \sim \overrightarrow{p_\theta}(X_{0,\Delta t,\dots,1})}\left[\log \frac{p_0(X_0)\overrightarrow{p_\theta}(X_{\Delta t,\dots,1} \mid X_0)}{\exp(-\mathcal{E}(X_1))\overleftarrow{p_\varphi}(X_{0,\dots,(T-1)\Delta t} \mid X_1)}\right] + \log Z. \end{aligned} \tag{10}$$

Here the log-normalising constant $\log Z$ does not affect the optimisation. Methods that propose to optimise this objective include [88, 77]. They use the reparametrisation trick to rewrite (10) as an expectation over the noises used in integration (the ξ_t in (4)). Although this scheme yields an unbiased estimator, it requires backpropagating though the simulation of the generation process. Alternatively, one can use second-moment divergences of the form

$$\mathbb{D}_{\tilde{p}}(p_0 \overrightarrow{p_\theta} \| p_{\text{target}} \overleftarrow{p_\varphi}) = \mathbb{E}_{X_{0,\Delta t,\dots,1} \sim \tilde{p}(X_{0,\Delta t,\dots,1})}\left[\log \frac{p_0(X_0)\overrightarrow{p_\theta}(X_{\Delta t,\dots,1} \mid X_0)}{\exp(-\mathcal{E}(X_1))\overleftarrow{p_\varphi}(X_{0,\dots,(T-1)\Delta t} \mid X_1)} + \log \hat{Z}\right]^2 \tag{11}$$

where $\tilde{p}$ is some full-support proposal distribution not necessarily equal to $\overrightarrow{p_\theta}$ and $\log \hat{Z}$ is a scalar. This scalar absorbs the (unknown) true normalising constant Z of p_{target}, allowing to use the unnormalised log-density $-\mathcal{E}(X_1)$ in place of $\log p_{\text{target}}(X_1) = -\mathcal{E}(X_1) - \log Z$ in the denominator. The scalar $\log \hat{Z}$ can be a learned parameter (yielding the loss called **trajectory balance** [**TB**; 41]) or analytically computed to minimise the loss on each batch of training trajectories (yielding the log-variance or **VarGrad** loss [60]). In either case, when the loss is minimised to 0, $\log \hat{Z}$ becomes equal to the true log-partition function $\log Z$.

Unlike the KL divergence (10), which is an expectation over a distribution depending on θ, second-moment divergences avoid backpropagating through sampling, allowing the use of off-policy exploration techniques to construct the proposal distribution $\tilde{p}$. On the other hand, when $\tilde{p} = p_\theta$, the expected gradient of $\mathbb{D}_{\tilde{p}}$ with respect to θ – *but not with respect to* φ – remarkably coincides with that of $\mathbb{D}_{\text{KL}}$ [42].

The divergence in (10) can be used for optimisation of $\overleftarrow{p_\varphi}$ in addition to $\overrightarrow{p_\theta}$. Notice that this is equivalent to log-likelihood maximisation with respect to φ, since the gradient of (10) takes the form:

$$\nabla_\varphi \mathbb{D}_{\text{KL}}(p_0 \overrightarrow{p_\theta} \| p_{\text{target}} \overleftarrow{p_\varphi}) = \nabla_\varphi \mathbb{E}_{X_{0,\dots,1} \sim \overrightarrow{p_\theta}(X_{0,\dots,1})}\left[-\log\left(\overleftarrow{p_\varphi}(X_{0,\dots,(T-1)\Delta t} \mid X_1)\right)\right] \tag{12}$$

$$= \nabla_\varphi \mathbb{E}_{X_{0,\dots,1} \sim \overrightarrow{p_\theta}(X_{0,\dots,1})}\left[\sum_{i=1}^{T} -\log \overleftarrow{p_\varphi}(X_{(i-1)\Delta t} \mid X_{i\Delta t})\right] \tag{13}$$

In the more general GFlowNet setting, such approach for learning the destruction process is called **trajectory likelihood maximisation** [**TLM**; 21].

B.3 Details on process parametrisations

We begin with the destruction process that is the reversal of time-discretised Brownian motion with fixed diffusion coefficient σ starting from $p_0 = \delta_0$, as considered by [88] and many later works. For this process, the transitions have the form:

$$\overrightarrow{p_\theta}(X_{t+\Delta t} \mid X_t) = \mathcal{N}(X_{t+\Delta t}; X_t + f_\theta(X_t, t)\,\Delta t,\ \sigma^2 \Delta t\, I_d), \tag{14}$$

$$\overleftarrow{p}(X_t \mid X_{t+\Delta t}) = \mathcal{N}\left(X_t; \frac{t}{t+\Delta t} X_{t+\Delta t},\ \frac{t}{t+\Delta t}\sigma^2 \Delta t\, I_d\right), \tag{15}$$

where $\overleftarrow{p}(x_0 \mid x_{\Delta t})$ is understood as Dirac $\delta_0(x_0)$. We use the following parametrisation for neural networks from (8):

$$\overrightarrow{p_\theta}(X_{t+\Delta t} \mid X_t) = \mathcal{N}(X_{t+\Delta t}; X_t + f_\theta(X_t, t)\,\Delta t,\ \mathrm{diag}(\gamma_\theta(X_t, t))\sigma^2 \Delta t\, I_d),$$
$$\gamma_\theta(X_t, t) = \exp\left\{C_1 \tanh\left(\mathrm{NN}^{(1)}_\theta(X_t, t)\right)\right\} \tag{16}$$

$$\overleftarrow{p_\varphi}(X_t \mid X_{t+\Delta t}) = \mathcal{N}\left(X_t; \mathrm{diag}(\alpha_\varphi(X_t, t))\,\frac{t}{t+\Delta t} X_{t+\Delta t},\ \mathrm{diag}(\beta_\varphi(X_t, t))\,\frac{t}{t+\Delta t}\sigma^2 \Delta t\, I_d\right),$$
$$\alpha_\varphi(X_t, t) = 1 + C_2 \tanh\left(\mathrm{NN}^{(2)}_\varphi(X_t, t)\right),\quad \beta_\varphi(X_t, t) = 1 + C_2 \tanh\left(\mathrm{NN}^{(3)}_\varphi(X_t, t)\right), \tag{17}$$

where the $\mathrm{NN}^{(i)}$ represent neural networks with d-dimensional outputs and C_1, C_2 are constants. Such a parametrisation guarantees that the generation process variance remains between $\frac{1}{C_1}$ and C_1 times that of the baseline one in (14), while the multiplicative deviations of the destruction process mean and variance from the basic ones in (15) are bounded between $1 - C_2$ and $1 + C_2$. We empirically found such a combination to work better than using the same parametrisation for both generation and destruction processes. This parametrisation does not impose any other constraints on the generation and destruction processes – they are no longer discretisations of SDEs defining absolutely continuous path space measures.

B.4 On soft RL equivalence.

In this section we discuss a fact that provides important motivation and context for some of the techniques mentioned in §2.4: it is possible to formulate the training of a diffusion sampler in discrete time as an RL problem with entropy regularisation (also called soft RL [51, 19]). Although, up to our knowledge, it was never directly stated in this form in previous literature on diffusion samplers, this fact is well-understood, and a number of works on diffusion samplers frame their training as a stochastic control task in continuous time [88, 79, 8]. The equivalence with soft RL in discrete time can be directly shown by combining the analysis of [73, 18], which show GFlowNet training is equivalent to a soft RL task, and [37, 64, 7], which show that discrete-time diffusion samplers are a special case of continuous GFlowNets.

This connection gives an important motivation for using such techniques as target networks [44], replay buffers [63], and various exploration methods, which are all well-studied in the RL literature, for the training of diffusion samplers. Finally, we mention that [21] analyses trainable destruction processes in GFlowNets from the RL perspective and theoretically and experimentally stipulates the use of techniques for enforcing stability in the training of the destruction process, such as controlled learning rates and target networks.

Below, we show that training a diffusion sampler with a finite number of steps and a fixed destruction process is equivalent to solving an entropy-regularized reinforcement learning (RL) problem.

We formalize the RL problem using a finite-horizon Markov Decision Process (MDP) [56, Chapter 4], defined as the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathsf{P}, \mathsf{r}, H, s_0)$, where $\mathcal{S}$ and $\mathcal{A}$ are measurable state and action spaces, $\mathsf{P}_h(\mathrm{d}s'|s, a)$ is a time-inhomogeneous transition kernel, $\mathsf{r}_h(s, a)$ is a time-dependent reward function with terminal reward $\mathsf{r}_H(s)$, H is the planning horizon, and s_0 is the initial state. We focus on deterministic MDPs, where the transition kernel is

$$\mathsf{P}_h(\mathrm{d}s' \mid s, a) = \delta_{\mathsf{T}_h(s,a)}(\mathrm{d}s')\,,$$

with $\mathsf{T}_h(s, a)$ a deterministic transition map and $\delta_y(\mathrm{d}x)$ the Dirac measure at y. The action space is equipped with a base measure $\mathrm{d}\mu(a)$ (e.g., the Lebesgue measure).

A time-inhomogeneous policy $\pi = \{\pi_h\}_{h=0}^{H-1}$ is a collection of conditional densities $\pi_h(a|s)$ with respect to $\mathrm{d}\mu(a)$. The corresponding entropy-regularized (or soft) value function [51, 19] is defined as

$$V^\pi_{\lambda,0}(s) = \mathbb{E}_\pi\left[\sum_{h=0}^{H-1} \left(\mathsf{r}_h(S_h, A_h) - \lambda \log \pi_h(A_h|S_h)\right) + \mathsf{r}_H(S_H) \,\middle|\, S_0 = s\right],$$

where $\lambda \geq 0$ is a regularization coefficient, and the expectation is over the trajectory induced by $A_h \sim \pi_h(\cdot|S_h)$ and $S_{h+1} = \mathsf{T}_h(S_h, A_h)$ for $h = 0, \ldots, H-1$.

We call a destruction process $\overleftarrow{p} = (\overleftarrow{p_t})_{t=0,\Delta t,\ldots,1}$ *regular* if $\overleftarrow{p_{\Delta t}}(\mathrm{d}x_0|x_{\Delta t}) = \delta_0(\mathrm{d}x_0)$ and $\overleftarrow{p_t}(\cdot|x_t)$ has a full support of $\mathbb{R}^d$ for all $t > \Delta t$. We use $\overleftarrow{p_{t+\Delta t}}(x_t|x_{t+\Delta t})$ as a corresponding density.

Theorem B.1. *Define* $\mathcal{E}(x)\colon \mathbb{R}^d \to \mathbb{R}$ *as an energy function, a target density* $p_{\text{target}}(x) = \exp(-\mathcal{E}(x))/Z$*, and a regular destruction process* $\overleftarrow{p} = (\overleftarrow{p_t})_{t=0,\Delta t,\dots,1}$.

Define a Markov decision process $\mathcal{M}_{\text{DS}}$ *with a state space equal to* $\mathcal{S} = \mathbb{R}^d$*, an action space equal to* $\mathcal{A} = \mathbb{R}^d$*, deterministic transition kernel defined by the following transition function* $\mathsf{T}_h(s,a) = a$*, a planning horizon* $H = T$*, and reward function* $\mathsf{r}_h(s,a) = \log \overleftarrow{p}_{h\cdot\Delta t}(a \mid s)$ *for* $h < H$*, with terminal reward* $\mathsf{r}_H(x) = -\mathcal{E}(x)$.

Then, for initial state $s_0 = 0$*, the soft value with* $\lambda = 1$ *in the MDP* $\mathcal{M}_{\text{DS}}$ *satisfies*

$$V^{\pi}_{\lambda=1,0}(s_0) = \log Z - \mathbb{D}_{\text{KL}}(p_0\overrightarrow{p_\pi} \| p_{\text{target}}\overleftarrow{p})\,,$$

where $\overrightarrow{p_\pi}$ *is a generation process that corresponds to a policy* π*:* $\overrightarrow{p_{\pi,t}}(x_{t+\Delta t} \mid x_t) = \pi_{t/\Delta t}(x_{t+\Delta t} \mid x_t)$*. As a result, the policy corresponding to the optimal policy* $\pi^\star_h$ *in the entropy-regularized MDP with* $\lambda = 1$ *provides a generation process that samples from the target distribution* $p_{\text{target}}(x) \propto \exp(-\mathcal{E}(x))$.

Proof. Let us start from the expression (10) for the reverse KL divergence between the corresponding generation and destruction processes:

$$\mathbb{D}_{\text{KL}}(p_0\overrightarrow{p_\pi} \| p_{\text{target}}\overleftarrow{p}) = \mathbb{E}_{X_{0,\Delta t,\dots,1}\sim\overrightarrow{p_\pi}(X_{0,\Delta t\dots,1})}\left[\log \frac{p_0(X_0)\overrightarrow{p_\pi}(X_{\Delta t\dots,1} \mid X_0)}{\exp(-\mathcal{E}(X_1))\overleftarrow{p}(X_{0,\dots,(T-1)\Delta t} \mid X_1)}\right] + \log Z.$$

Next, we use a chain rule for the probability distribution to study the first term in the expansion above:

$$\mathbb{E}_{X_{0,\Delta t,\dots,1}\sim\overrightarrow{p_\pi}(X_{0,\Delta t\dots,1})}\left[\sum_{i=0}^{T-1} \log \overrightarrow{p_{\pi,i\Delta t}}(X_{(i+1)\Delta t} \mid X_{i\Delta t}) - \log \overleftarrow{p_{(i+1)\Delta t}}(X_{i\Delta t} \mid X_{(i+1)\Delta t}) + \mathcal{E}(X_1)\right].$$

To show the equivalence with the corresponding soft RL definition of the value, we rename variables as follows: $T \mapsto H$, $i \mapsto h$, and $S_{h+1} = A_h = X_{(i+1)\Delta t}$, apply a definition of a reward function and of the generation process induced by policy:

$$\mathbb{D}_{\text{KL}}(p_0\overrightarrow{p_\pi} \| p_{\text{target}}\overleftarrow{p}) - \log Z = \mathbb{E}_\pi\left[\sum_{h=0}^{H-1} (\log \pi_h(A_h|S_h) - \mathsf{r}_h(S_h,A_h)) - \mathsf{r}_H(S_H)\right].$$

In the right-hand side of the equation we see exactly a negative value function with $\lambda = 1$. □

The same result was obtained by [73] in the GFlowNet framework. In addition, we would like to add that this perspective allows to treat the Trajectory Balance loss as a specific instance of Path Consistency Learning [47], as it was shown for GFlowNets in [18] (see also [75] for a discussion in the context of RL-based fine-tuning of diffusion models).

C Techniques for stability

C.1 Implementation details of techniques for stability

In this section, we describe the implementation details of stability techniques and discuss unsuccessful approaches for destruction process training.

Shared backbone. Our architecture is built upon the one used in [64], uses time and state encoders, a shared backbone and 2 different final layers for generation and destruction policy modelling. We use an MLP with GELU [23] activation for the shared backbone. For experiments on Manywell, Distorted Manywell, and GAN, this MLP has 4 layers, and for other energies it has 2.

Separate optimisers. The optimiser for generation policy updates parameters of time and state encoders, the backbone, and the final layer modelling the generation process. The optimiser for destruction policy updates the parameters of the time and state encoders, the backbone, and the final layer modelling the destruction process. For both optimisers, we use Adam [35] with standard parameters and weight decay of 10^{-7}.

Target network. Using a target network introduces a specific coefficient τ, which defines the update speed of the target network. On each iteration, the target weights are updated by:

$$\overline{\theta} = (1-\tau)\overline{\theta} + \tau\theta, \tag{18}$$

where $\overline{\theta}$ are the target network weights. If τ is set to 0, the target network is the current policy network, while higher τ leads to a slower evolution of the target network. For the generation policy

Table 3: Different configurations compared by ELBO, EUBO and 2-Wasserstein between generated and ground truth samples on 40GMM. Mean and std over 5 runs are specified.

Ablation	Setting	ELBO (↑)	EUBO (↓)	2-Wasserstein (↓)
(Optimal configuration)	$\text{TB}_{\theta,\varphi}$, shared backbone, separate optimisers, $\text{lr}_\varphi = \text{lr}_\theta$, replay ratio = 3, target network	$-\mathbf{1.89}_{\pm 0.10}$	$\mathbf{5.06}_{\pm 0.35}$	$18.71_{\pm 0.63}$
Fixed variance	$\text{TB}_{\theta,\varphi}$, fixed generation and destruction var.	$-2.49_{\pm 0.05}$	$101.84_{\pm 11.23}$	$30.15_{\pm 2.30}$
Learning only generation	TB_θ, fixed generation var. TB_θ, learnt generation var.	$-2.47_{\pm 0.06}$ $-3.5_{\pm 0.11}$	$105.1_{\pm 27.11}$ $37.02_{\pm 45.42}$	$32.56_{\pm 2.3}$ $20.37_{\pm 6.14}$
Models and optimisers	Separate backbones Single optimizer	$-6.03_{\pm 1.15}$ $-2.08_{\pm 0.15}$	$26.71_{\pm 2.70}$ $13.05_{\pm 4.95}$	$28.29_{\pm 0.49}$ $20.40_{\pm 0.71}$
Learning rates	$\text{lr}_\varphi = 0.10 \times \text{lr}_\theta$ $\text{lr}_\varphi = 0.01 \times \text{lr}_\theta$	$-2.06_{\pm 0.20}$ $-2.69_{\pm 0.94}$	$6.55_{\pm 3.48}$ $61.43_{\pm 30.20}$	$\mathbf{16.79}_{\pm 1.71}$ $21.77_{\pm 3.68}$
Replay ratio	Replay ratio = 1 Replay ratio = 2 Replay ratio = 5 Replay ratio = 9	$-1.96_{\pm 0.13}$ $-2.28_{\pm 1.05}$ $\mathbf{-1.91}_{\pm 0.18}$ $-3.43_{\pm 1.59}$	$18.59_{\pm 11.72}$ $23.53_{\pm 15.92}$ $6.08_{\pm 1.46}$ $21.24_{\pm 21.20}$	$20.86_{\pm 1.10}$ $\mathbf{16.74}_{\pm 1.10}$ $\mathbf{16.74}_{\pm 1.10}$ $\mathbf{16.79}_{\pm 1.71}$
Target network	No target network	$-2.07_{\pm 0.09}$	$10.36_{\pm 2.02}$	$20.13_{\pm 1.10}$

network, the second moment loss in (11) transforms into:

$$\mathbb{D}^{\text{gen.}}_{\tilde{p}}(p_0\overrightarrow{p_\theta}\|p_{\text{target}}\overleftarrow{p_{\overline{\varphi}}}) = \mathbb{E}_{X_{0,\Delta t,\ldots,1}\sim\tilde{p}(X_{0,\Delta t\ldots,1})}\left[\log\frac{p_0(X_0)\overrightarrow{p_\theta}(X_{\Delta t\ldots,1}\mid X_0)}{\exp(-\mathcal{E}(X_1))\overleftarrow{p_{\overline{\varphi}}}(X_{0,\ldots,(T-1)\Delta t}\mid X_1)} + \log\hat{Z}\right]^2, \tag{19}$$

where $\overline{\varphi}$ is the frozen destruction target network weights, and other variables are the same as in (11).

Similarly, the second moment loss for the destruction policy transforms into:

$$\mathbb{D}^{\text{destr.}}_{\tilde{p}}(p_0\overrightarrow{p_\theta}\|p_{\text{target}}\overleftarrow{p_\varphi}) = \mathbb{E}_{X_{0,\Delta t,\ldots,1}\sim\tilde{p}(X_{0,\Delta t\ldots,1})}\left[\log\frac{p_0(X_0)\overrightarrow{p_{\overline{\theta}}}(X_{\Delta t\ldots,1}\mid X_0)}{\exp(-\mathcal{E}(X_1))\overleftarrow{p_\varphi}(X_{0,\ldots,(T-1)\Delta t}\mid X_1)} + \log\hat{Z}\right]^2, \tag{20}$$

where $\overline{\theta}$ is the frozen generation target generation network, and other variables are as in (11).

Prioritised experience replay. We use the implementation of PER [63] from `torchrl` library [11]. We set the temperature parameter α to 1.0 and the importance sampling correction coefficient to 0.1 (similar parameter values were used in [73]).

Better exploration in off-policy methods. We use the existing techniques proposed by [37, 64] to facilitate exploration during training. We use a replay buffer of terminal states updated by Langevin dynamics (as studied by [64]), that is used to sample trajectories for training via the destruction process. We also sample trajectories from the current generation policy, but with increased variance on each step, with the added variance annealed to zero over the first 10 000 iterations (similar to the techniques studied by [42, 37]).

Other considerations. In addition to the techniques discussed in §2.4 we also present design choices that we tried in our experiments, but which turned out to be unsuccessful. We share them to offer deeper intuition behind the development of our final methodology:

- **Different parametrisation.** We tried to predict destruction variance in the log-scale in the same way as the generation variance (8):

$$\beta'_\varphi(X_t, t) = \exp\left\{C_1 \tanh\left(\text{NN}^{(2)}_\theta(X_t, t)\right)\right\}, \tag{21}$$

  but this parametrisation caused rapid fluctuations in the destruction policy and convergence was dramatically slower than with the parametrisation in (9).
- **Linear annealing.** In experiments on synthetic tasks, we set the constant C_2 in (9) to 0.9. We also tried to increase C_2 linearly from 0 to 0.9 over the duration of training. We initially thought this to be efficient since both generation and destruction processes are less stable in the beginning of training when modes are unexplored.

C.2 Ablation study

In this subsection, we discuss the results of the ablation study to test the design choices outlined in §2.4. The numerical results are presented in Table 3.

Parametrisation, optimisers, learning rates. As stated in §2.4, we experiment with using a shared backbone for the generation and destruction processes. We find that using a shared backbone drastically increases the quality of the sampler. Moreover, even though the backbone is shared between two networks, it is optimal to use separate optimisers for the two processes.

We empirically find that the performance of samplers is sensitive to learning rates, and thus they must be carefully tuned for each environment. We use equal learning rates for θ and φ for GMM distributions. However, for more complex environments the destruction policy learning rate must be smaller than that of the generation policy. For instance, in Hard Funnel, a 10^3 times smaller learning rate for φ is optimal, and for Manywell, the optimal ratio is 10^4 or 10^5.

Target network and replay buffer. The best replay ratio in our setup is 2, and we use this value for all our experiments. Moreover, using target networks increases the stability of the training and the final quality of the sampler.

D Extended results and findings

D.1 Results on synthetic tasks

Learned generation variances improve sampling. Learnable variance of the generation process significantly improves the performance of the sampler, especially when the number of generation steps is small. As shown in Fig. 1, models with learnable generation variance dramatically outperform ones with fixed variance: on some energies, learned-variance samplers with as few as 5 generation steps outperform 20-step samplers with fixed variance. Second, the importance of trainable variance is clearly observable in complex environments or environments with high distortions (Figs. 1 and 4). If the variance is fixed, the sampler is likely to struggle with correctly sampling narrow modes: in particular, the addition of noise of variance $\sigma^2 \Delta t$ on the last step imposes a smoothness constraint on the modelled distribution. Conversely, learning the magnitude of added noise allows to capture the shape of such modes.

Off-policy losses are superior to differentiable simulation. The off-policy TB loss with exploratory behaviour policy is on par with or better than the PIS loss for learning the generation process in all cases, as shown in the Appendix F. This is consistent with findings in past work, such as [59, 64, 34]. Extending these findings, we find that this improvement is maintained when generation variances and the destruction process are learned.

Learning the destruction process is beneficial. Learning the destruction process yields an improvement over models with fixed destruction process and learned generation variance on all tasks, although the improvement is often less pronounced when the number of sampling steps is large (*e.g.*, on the Funnel densities in Fig. 1), presumably because the reverse of a fixed destruction process is better modelled by Gaussian transitions when the number of steps is large.

TB is preferable to TLM for learning destruction. Extending the results of [21] in discrete cases, we find that training the destruction process with the TLM loss is typically superior to TB when the number of steps is small. However, the TLM loss is unstable and often leads to divergent training when the number of steps is large (see Appendix F). We note that, even at optimality, the TLM loss gradient has nonzero variance, while the TB loss gradient is zero for all trajectories at the global optimum, which may explain TB's greater stability.

D.2 Scalability: Sampling in GAN latent space for conditional image generation

To validate our main findings in a high-dimensional setting, we consider the setup proposed in [81]. Let $g_\psi : \mathbb{R}^{d_{\text{latent}}} \to \mathbb{R}^{d_{\text{data}}}$ be a pretrained GAN generator [20], and $r(x, y)$ be some positive-valued function operating on data points $x \in \mathbb{R}^{d_{\text{data}}}$ and conditions y. The task is to sample latent vectors z from the distribution defined by the energy $\mathcal{E}(z) = -\log p_{\text{prior}}(z) - \beta \log r(g_\psi(z), y)$, where p_{prior} is $\mathcal{N}(0, I)$. The decoded samples $g_\psi(z)$ then follow the posterior distribution, proportional to the product of the GAN image prior and the tempered constraint $r(x, y)^\beta$.

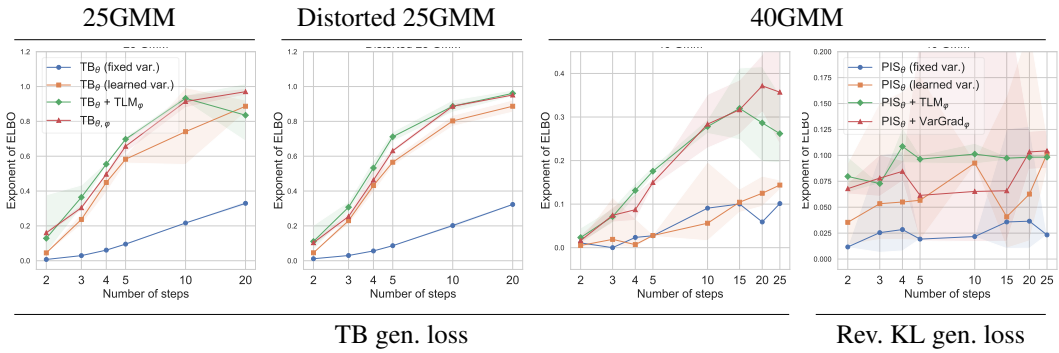

Figure 3: We compare performance of TB_θ (fixed var.), TB_θ (learned var.), TB_θ+TLM_φ, $\mathrm{TB}_{\theta,\varphi}$ on three GMM targets. The rightmost plot shows PIS-like generation process objectives on **40GMM**. Results are compared using ELBO. Mean and std over 3 seeds, collapsed runs excluded.

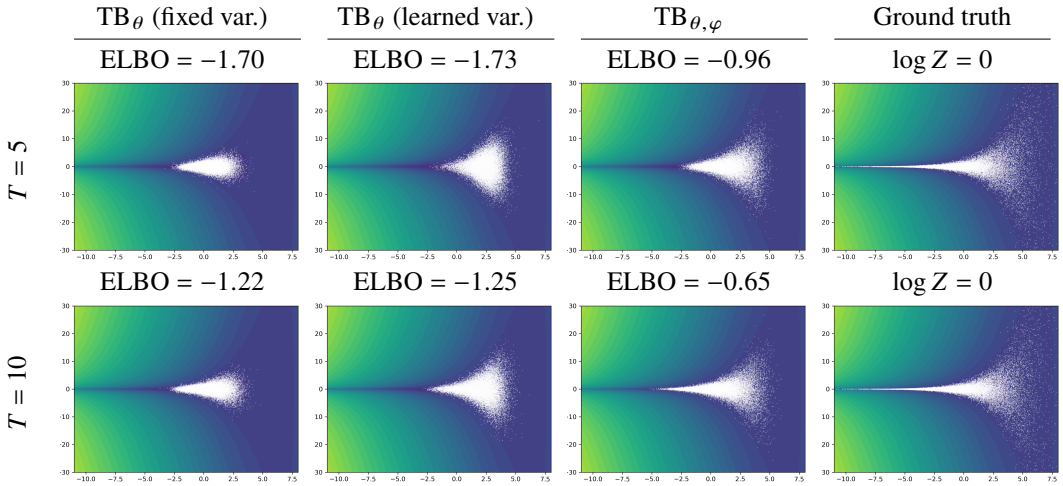

Figure 4: First two dimensions of target energy and samples from diffusion samplers trained on the Hard Funnel energy. Samplers with fixed destruction process, and especially those with fixed generation process variance, struggle to fit the narrow tails accurately.

In our experiment we use StyleGAN3 [31] trained on the 256×256 FFHQ dataset [32] with $d_{\text{latent}} = 512$. We take y to be a text prompt and $\log r(x, y)$ to be the ImageReward score [82]; thus our aim is to sample latents that produce faces aligned with the specified text prompt. We use the same value of $\beta = 100$ as in [81]. Note that the GAN itself is unconditional, and text-conditioning is only achieved via sampling from the distribution defined through ImageReward. An optimal model should trade off between high reward (satisfying the prompt) and high diversity (modelling all modes).

We train diffusion samplers with only $T = 5$ sampling steps to sample from the specified target distribution in $\mathbb{R}^{d_{\text{latent}}}$, comparing the approach from [81] (TB_θ with fixed generation variances) to $\mathrm{TB}_{\theta,\varphi}$ with learnable destruction process and generation variances. In addition to ELBO, we report average ImageReward score and diversity, measured as average cosine distance of CLIP [57] embeddings for 128 generated images. (Note that EUBO and 2-Wasserstein cannot be computed here since we have no access to ground-truth samples.) We train separate models across 7 different prompts and find improvements in ELBO and ImageReward on 5 of them, similar performance on 1, and slight degradation in 1. A representative example of the improvement is shown in Fig. 2. Average metrics are reported in Table 2. For further details see Appendix E.4.

E Experiment details

E.1 Definition of energies

We present contour levels of 2-dimensional Gaussian mixtures in Figure 5.

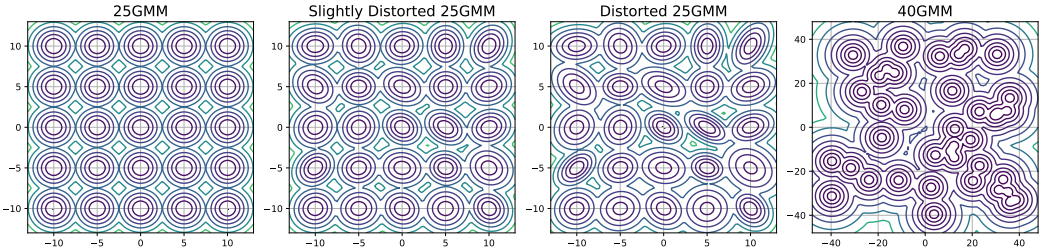

Figure 5: Contour levels of 2-dimensional Gaussian mixture densities used in the experiments.

25GMM. This is a mixture of 25 different Gaussians in a 2-dimensional space. Each component is Gaussian with variance 0.3. The means are arranged on a grid given by the Cartesian product $\{-10, -5, 0, 5, 10\} \times \{-10, -5, 0, 5, 10\}$.

Slightly Distorted 25GMM. This environment is obtained by a slight modification of 25GMM. The means are located in the same positions, but we distort the initial variance matrices with the following rule:

$$C_i = \left(\begin{bmatrix} \sqrt{0.3} & 0 \\ 0 & \sqrt{0.3} \end{bmatrix} + d \begin{bmatrix} \xi_{i1} & \xi_{i2} \\ \xi_{i3} & \xi_{i4} \end{bmatrix} \right)^\top \cdot \left(\begin{bmatrix} \sqrt{0.3} & 0 \\ 0 & \sqrt{0.3} \end{bmatrix} + d \begin{bmatrix} \xi_{i1} & \xi_{i2} \\ \xi_{i3} & \xi_{i4} \end{bmatrix} \right), \tag{22}$$

where C_i is the covariance matrix of the i-th mode, $\xi_{ij} \sim \mathcal{N}(0, 1)$, and d is set to 0.05. To achieve a fair comparison, we sample the random variables with a predefined seed 42, hence all algorithms are compared on the same distribution.

Distorted 25GMM. The same as Slightly Distorted 25GMM, but with d equal 0.1.

125GMM. The same as 25GMM, but in $\mathbb{R}^3$ with means at $\{-10, -5, 0, 5, 10\}^3$.

40GMM. This distribution is taken from [43]. It consists of 40 equally weighted mixture components with components sampled as:

$$\pi_k(x) = \mathcal{N}(x; \mu_k, I)$$
$$\mu_k \sim \mathcal{U}(-40, 40).$$

Easy/Hard Funnel. The funnel distribution serves as a classical benchmark in the evaluation of sampling methods. It is defined over a ten-dimensional space, where the first variable, x_0, is drawn from a normal distribution with mean 0 and variance 1 (Easy Funnel) or 9 (Hard Funnel), $x_0 \sim \mathcal{N}(0, 1)$. Given x_0, the remaining components $x_{1:9}$ follow a multivariate normal distribution with mean vector zero and covariance matrix $\exp(x_0)I$, where I denotes the identity matrix. This conditional relationship is expressed as $x_{1:9} \mid x_0 \sim \mathcal{N}(0, \exp(x_0)I)$.

Manywell. The distribution is defined over a 32-dimensional space and is constructed as the product of 16 identical 2-dimensional double-well distributions. Each of these two-dimensional components is governed by a potential function, $\mu(x_1, x_2)$, given by $\mu(x_1, x_2) = \exp\left(-x_1^4 + 6x_1^2 + 0.5x_1 - 0.5x_2^2\right)$.

Distorted Manywell. We modify the Manywell potential function for each 2-dimensional components:

$$\mu(x_{2i-1}, x_{2i}) = \exp\left(-a_{i,1}x_{2i-1}^4 + 6a_{i,2}x_{2i-1}^2 + 0.5a_{i,3}x_{2i-1} - 0.5a_{i,4}x_{2i}^2\right),$$

where $a_{i,j} \sim \mathcal{U}(0.75, 1.25)$.

E.2 Metrics

The ELBO and EUBO metrics are defined as follows:

$$\text{ELBO} = \mathbb{E}_{X_{0,\ldots,1} \sim p_0(X_0)\overrightarrow{p_\theta}(X_{0,\Delta t\ldots,1})} \left[\log \frac{\exp(-\mathcal{E}(X_1))\overleftarrow{p_\varphi}(X_{0,\ldots,(T-1)\Delta t} \mid X_1)}{p_0(X_0)\overrightarrow{p_\theta}(X_{\Delta t\ldots,1} \mid X_0)} \right]$$

$$\text{EUBO} = \mathbb{E}_{X_{0,\ldots,1} \sim p_{\text{target}}(X_1)p_\varphi(X_{0,\ldots,(T-1)\Delta t} \mid X_1)} \left[\log \frac{\exp(-\mathcal{E}(X_1))\overleftarrow{p_\varphi}(X_{0,\ldots,(T-1)\Delta t} \mid X_1)}{p_0(X_0)\overrightarrow{p_\theta}(X_{\Delta t\ldots,1} \mid X_0)} \right]$$

Both expectations are estimated using 2048 Monte Carlo samples. For the 2-Wasserstein distance (W_2^2), we also use 2048 ground truth and 2048 generated samples.

E.3 Synthetic tasks training details

Here we describe the chosen hyperparameters for the final results across synthetic tasks in §3.1. We train all samplers for 25 000 iterations. The diffusion rate σ^2 is set to 5 for experiments with Gaussian mixtures and to 1 for other energies. The batch size is 512. We apply zero initialisation for the final layers of the neural networks to obtain a uniform output in the beginning of the training. Additionally, we perform clipping on the output of the policy network by 10^{-4}. We also apply gradient clipping with value 200. The target neural network update speed τ is set 0.05.

We set C_1 in (8) and C_2 in (9) to 4.0 and 0.9 respectively.

For all experiments, we set lr_θ (learning rate for generation policy neural network) to 10^{-3}. The normalisation constant $\log Z$ is trained with learning rate 10^{-1}. We tune lr_φ (learning rate for destruction policy neural network) specifically for each energy and find that optimal lr_φ is always less than or equal to lr_θ. For the samplers trained with TB loss, we set lr_φ to lr_θ in experiments with 25GMM and 125GMM, choose the optimal lr_φ between $\{\text{lr}_\theta, 10^{-1} \times \text{lr}_\theta\}$ for 40GMM, set lr_φ to lr_θ for Easy Funnel and to $10^{-3} \times \text{lr}_\theta$ for Hard Funnel, we set lr_φ to $10^{-5} \times \text{lr}_\theta$ for $T = 5$ and to $10^{-4} \times \text{lr}_\theta$ for $T = 10$ and $T = 20$ in Manywell and Distorted Manywell. For PIS, we choose optimal lr_φ between $\{\text{lr}_\theta, 10^{-1} \times \text{lr}_\theta, 10^{-2} \times \text{lr}_\theta\}$ in experiments with Gaussian mixtures. We use an exponential learning rate schedule, which multiplies learning rate by γ after each on-policy gradient step. We set γ to 0.99988 in experiments with 25GMM and 40GMM and to 0.9999 in other tasks.

For off policy TB, we set the replay ratio to 2. We set the size of the experience replay buffer to 5000 and the size of the buffer used in local search to 600 000. All hyperparameters for local search are taken from [64]. We use exploration factor of 0.3 for Gaussian mixtures, 0.2 for Easy Funnel and Hard Funnel, and 0.1 for Manywell and Distorted Manywell. We note that we use replay buffers and off-policy exploration in all methods and configurations that allow for off-policy training to ensure a fair comparison.

We consider the architecture from [64], but we stack the time encoding with the state encoding rather than summing them. We set state, time, and hidden dimension sizes to 64 across all environments with the except for Manywell and Distorted Manywell, where we set these values to 256.

We use a harmonic discretisation scheme for all mixtures of Gaussians since this time discretisation leads to better sampling quality compared to uniform. It partitions the time space unevenly, making steps large around $t = 0$ and smaller closer to $t = 1$. The code for harmonic discretisation is presented in Appendix E.3. For other energies, we apply uniform time discretization.

All models in this section were trained on CPUs. Our implementations are based upon the published code of [64].

```
def harmonic_discretizer(batch_size: int, trajectory_length: int):
    step_sizes = 1 / arange(1, trajectory_length + 1)
    sum_step_sizes = sum(step_sizes)
    step_proportions = step_sizes / sum_step_sizes
    split_points = cumsum(step_proportions)
    return concatenate([0.0, split_points])
```

E.4 GAN latent sampler training details

We use similar setup and hyperparameters to the ones described in Appendix E.3, with a number of differences. We use smaller values of $C_1 = 1.0$ and $C_2 = 0.1$, which we found to improve the training stability and sampling quality in this task. We also use a smaller batch size of 128 and larger hidden size of 1024 for the MLP network (note that [81] uses a variant of UNet architecture [61], while we use a variant of the architecture from [64], see Appendix E.3). We set $\sigma^2 = 1$, which matches the variance of the prior. We set $\text{lr}_\theta = 10^{-3}$, $\text{lr}_\varphi = 0.2 \times \text{lr}_\theta$, $\gamma = 0.9999$. The exploration factor is set to 0.1 and the replay ratio is set to 5. We found it beneficial to set the target neural network update speed τ to a higher value of 0.15. We train all samplers for 20 000 iterations.

We note that replay buffers and off-policy exploration are used both for the baseline and for our approach to ensure a fair comparison. We do not use the local search method proposed in [64] as it requires access to gradients of the target energy function, which would require costly differentiation through the GAN generator and ImageReward. Thus all models considered in this experiment require access only to $\mathcal{E}(z)$.

For GAN latent space sampling experiments we used NVIDIA V100 GPUs. Our implementations are based upon the published code of [64], as well as the official implementations of [31, 57, 82].

Extending the results from Table 2, Fig. 6 depicts metric differences across all prompts utilized in this experiment.

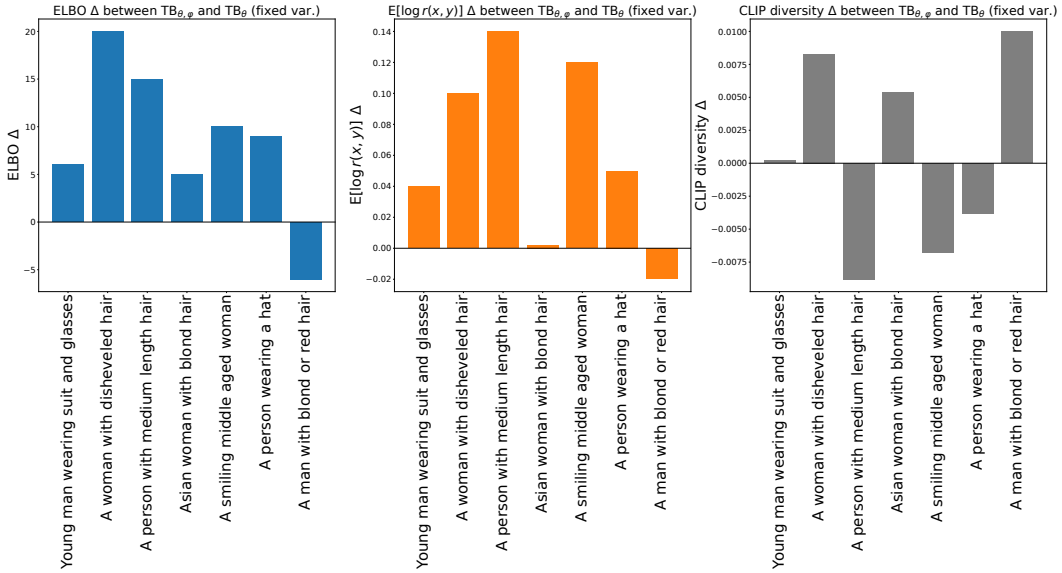

Figure 6: FFHQ text-conditional latent sampling results across different prompts. The figure depicts the difference in metrics between TB$_{\theta,\varphi}$ (ours) and TB$_\theta$ (fixed var.). *Left:* ELBO, *middle:* average ImageReward, *right:* CLIP diversity. Our method improves reward and ELBO in most prompts. Differences in CLIP diversity are minor (less than 0.01 for all prompts), and on average both methods show the same diversity when rounded up to two decimals.

F Supplementary tables

See Tables 4 to 8 on the following pages for full results.

Table 4: Comparison of 4 algorithms by ELBO, EUBO, and 2-Wasserstein between generated and ground truth samples with varying number of discretisation steps on 125GMM. Mean and std over 3 runs are specified.

ELBO (↑)

Energy ↓	Method ↓ Steps →		$T = 2$	$T = 3$	$T = 4$	$T = 5$	$T = 10$	$T = 20$
125GMM	TB_θ	(fixed var.)	$-7.36_{\pm 0.09}$	$-5.32_{\pm 0.01}$	$-4.29_{\pm 0.03}$	$-3.64_{\pm 0.06}$	$-2.39_{\pm 0.03}$	$-1.72_{\pm 0.01}$
	TB_θ + TLM_φ	(fixed var.)	$-7.33_{\pm 0.13}$	$-5.38_{\pm 0.09}$	$-3.76_{\pm 0.15}$	$-2.88_{\pm 0.10}$	$-1.22_{\pm 0.06}$	$-0.65_{\pm 0.04}$
	$\text{TB}_{\theta,\varphi}$	(fixed var.)	$-8.21_{\pm 0.13}$	$-5.51_{\pm 0.08}$	$-3.96_{\pm 0.02}$	$-3.13_{\pm 0.05}$	$-1.29_{\pm 0.05}$	$-0.70_{\pm 0.08}$
	TB_θ	(learnt var.)	$-4.92_{\pm 0.16}$	$-2.27_{\pm 0.06}$	$-1.31_{\pm 0.06}$	$-0.88_{\pm 0.08}$	$-0.36_{\pm 0.02}$	$-0.24_{\pm 0.03}$
	TB_θ + TLM_φ	(learnt var.)	$-3.82_{\pm 1.11}$	$-1.67_{\pm 0.10}$	$-1.31_{\pm 0.02}$	$-0.80_{\pm 0.10}$	$-0.34_{\pm 0.05}$	$-0.31_{\pm 0.24}$
	$\text{TB}_{\theta,\varphi}$	(learnt var.)	$-5.34_{\pm 1.09}$	$-2.06_{\pm 0.06}$	$-1.31_{\pm 0.13}$	$-0.86_{\pm 0.08}$	$-0.27_{\pm 0.09}$	$-0.06_{\pm 0.00}$
	PIS_θ	(fixed var.)	$-6.94_{\pm 0.32}$	$-5.34_{\pm 0.63}$	$-4.75_{\pm 0.73}$	$-3.68_{\pm 0.13}$	$-4.85_{\pm 0.13}$	$-2.83_{\pm 0.12}$
	PIS_θ	(learnt var.)	$-3.18_{\pm 0.59}$	$-2.75_{\pm 0.26}$	$-2.09_{\pm 0.04}$	$-2.05_{\pm 0.04}$	$-1.87_{\pm 0.01}$	$-2.06_{\pm 0.08}$
	PIS_θ + TLM_φ	(learnt var.)	$-1.06_{\pm 0.34}$	$-1.29_{\pm 0.15}$	$-1.31_{\pm 0.13}$	$-1.05_{\pm 0.28}$	$-1.20_{\pm 0.20}$	$-1.80_{\pm 0.31}$
	PIS_θ + VarGrad_φ	(learnt var.)	$-4.83_{\pm 0.00}$	$-4.83_{\pm 0.00}$	$-2.25_{\pm 0.25}$	$-2.04_{\pm 0.01}$	$-2.13_{\pm 0.47}$	$-2.16_{\pm 0.54}$

EUBO (↓)

Energy ↓	Method ↓ Steps →		$T = 2$	$T = 3$	$T = 4$	$T = 5$	$T = 10$	$T = 20$
125GMM	TB_θ	(fixed var.)	$10.71_{\pm 0.33}$	$9.51_{\pm 0.08}$	$8.82_{\pm 0.14}$	$8.15_{\pm 0.15}$	$6.04_{\pm 0.32}$	$4.86_{\pm 0.88}$
	TB_θ + TLM_φ	(fixed var.)	$13.00_{\pm 0.55}$	$6.08_{\pm 0.12}$	$4.46_{\pm 0.21}$	$3.03_{\pm 0.09}$	$1.26_{\pm 0.16}$	$0.63_{\pm 0.02}$
	$\text{TB}_{\theta,\varphi}$	(fixed var.)	$14.23_{\pm 0.46}$	$10.75_{\pm 0.27}$	$6.18_{\pm 0.13}$	$4.89_{\pm 0.22}$	$1.27_{\pm 0.08}$	$0.69_{\pm 0.08}$
	TB_θ	(learnt var.)	$2.19_{\pm 0.07}$	$1.15_{\pm 0.04}$	$0.77_{\pm 0.03}$	$0.56_{\pm 0.02}$	$0.27_{\pm 0.01}$	$0.18_{\pm 0.03}$
	TB_θ + TLM_φ	(learnt var.)	$1.58_{\pm 0.36}$	$1.01_{\pm 0.02}$	$0.81_{\pm 0.03}$	$0.52_{\pm 0.06}$	$0.30_{\pm 0.06}$	$0.50_{\pm 0.53}$
	$\text{TB}_{\theta,\varphi}$	(learnt var.)	$2.06_{\pm 0.31}$	$1.15_{\pm 0.10}$	$0.72_{\pm 0.03}$	$0.51_{\pm 0.04}$	$0.21_{\pm 0.05}$	$0.06_{\pm 0.00}$
	PIS_θ	(fixed var.)	$11.68_{\pm 0.69}$	$10.71_{\pm 0.70}$	$11.68_{\pm 0.61}$	$11.68_{\pm 0.61}$	$12.47_{\pm 0.44}$	$12.83_{\pm 0.20}$
	PIS_θ	(learnt var.)	$11.70_{\pm 1.57}$	$12.70_{\pm 0.23}$	$12.81_{\pm 0.15}$	$12.70_{\pm 0.15}$	$12.57_{\pm 0.20}$	$11.61_{\pm 0.57}$
	PIS_θ + TLM_φ	(learnt var.)	$4.98_{\pm 0.86}$	$4.53_{\pm 0.36}$	$4.01_{\pm 0.39}$	$3.30_{\pm 1.56}$	$3.67_{\pm 0.87}$	$4.37_{\pm 0.44}$
	PIS_θ + VarGrad_φ	(learnt var.)	$8.80_{\pm 2.96}$	$10.50_{\pm 3.13}$	$11.89_{\pm 1.12}$	$11.42_{\pm 1.41}$	$7.88_{\pm 3.26}$	$7.60_{\pm 3.75}$

2-Wasserstein (↓)

Energy ↓	Method ↓ Steps →		$T = 2$	$T = 3$	$T = 4$	$T = 5$	$T = 10$	$T = 20$
125GMM	TB_θ	(fixed var.)	$8.12_{\pm 0.12}$	$7.61_{\pm 0.15}$	$7.26_{\pm 0.14}$	$6.95_{\pm 0.10}$	$6.28_{\pm 0.03}$	$5.65_{\pm 0.23}$
	TB_θ + TLM_φ	(fixed var.)	$7.76_{\pm 0.35}$	$6.49_{\pm 0.15}$	$5.85_{\pm 0.04}$	$5.25_{\pm 0.13}$	$3.48_{\pm 0.12}$	$2.49_{\pm 0.06}$
	$\text{TB}_{\theta,\varphi}$	(fixed var.)	$7.68_{\pm 0.16}$	$6.80_{\pm 0.18}$	$6.01_{\pm 0.11}$	$5.67_{\pm 0.17}$	$3.52_{\pm 0.09}$	$2.63_{\pm 0.12}$
	TB_θ	(learnt var.)	$3.46_{\pm 0.09}$	$3.09_{\pm 0.13}$	$2.95_{\pm 0.20}$	$2.74_{\pm 0.04}$	$2.43_{\pm 0.05}$	$1.93_{\pm 0.05}$
	TB_θ + TLM_φ	(learnt var.)	$1.97_{\pm 0.16}$	$2.82_{\pm 0.15}$	$2.60_{\pm 0.14}$	$2.23_{\pm 0.31}$	$2.27_{\pm 0.36}$	$2.37_{\pm 1.04}$
	$\text{TB}_{\theta,\varphi}$	(learnt var.)	$2.54_{\pm 0.46}$	$2.95_{\pm 0.29}$	$2.47_{\pm 0.08}$	$2.21_{\pm 0.19}$	$1.85_{\pm 0.11}$	$1.84_{\pm 0.09}$
	PIS_θ	(fixed var.)	$7.60_{\pm 0.14}$	$7.01_{\pm 0.36}$	$7.32_{\pm 0.54}$	$6.91_{\pm 0.48}$	$7.19_{\pm 0.31}$	$7.94_{\pm 0.47}$
	PIS_θ	(learnt var.)	$7.28_{\pm 1.92}$	$7.32_{\pm 0.59}$	$5.82_{\pm 0.06}$	$5.91_{\pm 0.14}$	$5.73_{\pm 0.07}$	$6.12_{\pm 0.26}$
	PIS_θ + TLM_φ	(learnt var.)	$3.25_{\pm 0.82}$	$4.61_{\pm 0.33}$	$4.96_{\pm 0.05}$	$4.24_{\pm 0.85}$	$5.01_{\pm 0.18}$	$5.34_{\pm 0.20}$
	PIS_θ + VarGrad_φ	(learnt var.)	$6.97_{\pm 1.23}$	$7.83_{\pm 1.06}$	$5.68_{\pm 0.07}$	$5.54_{\pm 0.04}$	$6.22_{\pm 1.09}$	$6.38_{\pm 1.20}$
	Ground Truth				$1.81_{\pm 0.11}$			

Table 5: Comparison of 4 algorithms by ELBO, EUBO and 2-Wasserstein between generated and ground truth samples with varying number of discretisation steps on 3 variants of 25GMM. Mean and std over 3 runs are specified.

ELBO (↑)

Energy ↓	Method ↓ Steps →		$T=2$	$T=3$	$T=4$	$T=5$	$T=10$	$T=20$
25GMM	TB_θ	(fixed var.)	$-4.92_{\pm 0.06}$	$-3.53_{\pm 0.05}$	$-2.80_{\pm 0.03}$	$-2.35_{\pm 0.04}$	$-1.53_{\pm 0.01}$	$-1.11_{\pm 0.02}$
	TB_θ + TLM_φ	(fixed var.)	$-4.75_{\pm 0.05}$	$-3.31_{\pm 0.05}$	$-2.39_{\pm 0.02}$	$-1.79_{\pm 0.05}$	$-0.70_{\pm 0.04}$	$-0.37_{\pm 0.03}$
	$\text{TB}_{\theta,\varphi}$	(fixed var.)	$-5.34_{\pm 0.11}$	$-3.76_{\pm 0.06}$	$-2.61_{\pm 0.04}$	$-2.01_{\pm 0.06}$	$-0.75_{\pm 0.05}$	$-0.37_{\pm 0.03}$
	TB_θ	(learnt var.)	$-3.10_{\pm 0.07}$	$-1.44_{\pm 0.07}$	$-0.80_{\pm 0.07}$	$-0.54_{\pm 0.02}$	$-0.30_{\pm 0.15}$	$-0.12_{\pm 0.02}$
	TB_θ + TLM_φ	(learnt var.)	$-2.05_{\pm 0.55}$	$-1.01_{\pm 0.09}$	$-0.59_{\pm 0.01}$	$-0.36_{\pm 0.01}$	$-0.07_{\pm 0.01}$	$-0.18_{\pm 0.10}$
	$\text{TB}_{\theta,\varphi}$	(learnt var.)	$-1.83_{\pm 0.04}$	$-1.19_{\pm 0.03}$	$-0.70_{\pm 0.01}$	$-0.42_{\pm 0.03}$	$-0.09_{\pm 0.02}$	$-0.03_{\pm 0.01}$
	PIS_θ	(fixed var.)	$-4.36_{\pm 0.04}$	$-3.23_{\pm 0.01}$	$-2.65_{\pm 0.01}$	$-2.36_{\pm 0.04}$	$-1.92_{\pm 0.09}$	$-1.64_{\pm 0.14}$
	PIS_θ	(learnt var.)	$-2.14_{\pm 0.81}$	$-1.45_{\pm 0.02}$	$-1.37_{\pm 0.01}$	$-1.31_{\pm 0.01}$	$-1.20_{\pm 0.01}$	$-1.34_{\pm 0.13}$
	PIS_θ + TLM_φ	(learnt var.)	$-1.13_{\pm 0.19}$	$-0.76_{\pm 0.04}$	$-0.82_{\pm 0.16}$	$-1.13_{\pm 0.01}$	$-1.01_{\pm 0.17}$	$-1.81_{\pm 0.30}$
	PIS_θ + VarGrad_φ	(learnt var.)	$-2.37_{\pm 0.00}$	$-1.75_{\pm 0.38}$	$-1.29_{\pm 0.01}$	$-1.18_{\pm 0.10}$	$-1.19_{\pm 0.01}$	$-1.13_{\pm 0.01}$

EUBO (↓)

Energy ↓	Method ↓ Steps →		$T=2$	$T=3$	$T=4$	$T=5$	$T=10$	$T=20$
25GMM	TB_θ	(fixed var.)	$7.01_{\pm 0.08}$	$5.91_{\pm 0.07}$	$5.33_{\pm 0.13}$	$5.07_{\pm 0.22}$	$4.11_{\pm 0.52}$	$2.43_{\pm 0.13}$
	TB_θ + TLM_φ	(fixed var.)	$7.25_{\pm 0.50}$	$3.57_{\pm 0.08}$	$2.33_{\pm 0.07}$	$1.64_{\pm 0.06}$	$0.67_{\pm 0.02}$	$0.35_{\pm 0.01}$
	$\text{TB}_{\theta,\varphi}$	(fixed var.)	$9.39_{\pm 0.19}$	$7.77_{\pm 0.96}$	$3.93_{\pm 0.66}$	$2.67_{\pm 0.45}$	$0.70_{\pm 0.06}$	$0.35_{\pm 0.01}$
	TB_θ	(learnt var.)	$1.41_{\pm 0.04}$	$0.74_{\pm 0.01}$	$0.47_{\pm 0.01}$	$0.35_{\pm 0.01}$	$0.90_{\pm 1.05}$	$0.09_{\pm 0.00}$
	TB_θ + TLM_φ	(learnt var.)	$0.99_{\pm 0.14}$	$0.56_{\pm 0.05}$	$0.36_{\pm 0.01}$	$0.24_{\pm 0.01}$	$0.07_{\pm 0.00}$	$1.87_{\pm 1.66}$
	$\text{TB}_{\theta,\varphi}$	(learnt var.)	$0.94_{\pm 0.03}$	$0.60_{\pm 0.02}$	$0.37_{\pm 0.02}$	$0.24_{\pm 0.00}$	$0.07_{\pm 0.01}$	$0.03_{\pm 0.00}$
	PIS_θ	(fixed var.)	$7.07_{\pm 0.23}$	$7.19_{\pm 0.22}$	$7.31_{\pm 0.54}$	$7.12_{\pm 0.20}$	$8.51_{\pm 0.06}$	$8.51_{\pm 0.06}$
	PIS_θ	(learnt var.)	$8.41_{\pm 0.20}$	$8.40_{\pm 0.06}$	$8.42_{\pm 0.04}$	$8.44_{\pm 0.03}$	$8.40_{\pm 0.16}$	$7.02_{\pm 0.05}$
	PIS_θ + TLM_φ	(learnt var.)	$6.03_{\pm 0.61}$	$5.41_{\pm 0.43}$	$3.45_{\pm 0.63}$	$5.50_{\pm 2.13}$	$4.25_{\pm 1.77}$	$7.61_{\pm 1.29}$
	PIS_θ + VarGrad_φ	(learnt var.)	$8.40_{\pm 0.06}$	$8.42_{\pm 0.08}$	$8.43_{\pm 0.08}$	$6.76_{\pm 1.25}$	$3.58_{\pm 0.49}$	$5.71_{\pm 2.01}$

2-Wasserstein (↓)

Energy ↓	Method ↓ Steps →		$T=2$	$T=3$	$T=4$	$T=5$	$T=10$	$T=20$
25GMM	TB_θ	(fixed var.)	$6.47_{\pm 0.05}$	$6.08_{\pm 0.07}$	$5.70_{\pm 0.02}$	$5.52_{\pm 0.08}$	$5.03_{\pm 0.17}$	$4.39_{\pm 0.01}$
	TB_θ + TLM_φ	(fixed var.)	$6.27_{\pm 0.09}$	$5.11_{\pm 0.10}$	$4.44_{\pm 0.06}$	$3.87_{\pm 0.04}$	$2.38_{\pm 0.11}$	$1.66_{\pm 0.07}$
	$\text{TB}_{\theta,\varphi}$	(fixed var.)	$6.36_{\pm 0.07}$	$5.74_{\pm 0.18}$	$4.75_{\pm 0.10}$	$4.32_{\pm 0.06}$	$2.56_{\pm 0.15}$	$1.77_{\pm 0.08}$
	TB_θ	(learnt var.)	$2.47_{\pm 0.03}$	$2.31_{\pm 0.10}$	$2.18_{\pm 0.06}$	$2.04_{\pm 0.10}$	$2.11_{\pm 0.90}$	$1.46_{\pm 0.30}$
	TB_θ + TLM_φ	(learnt var.)	$1.37_{\pm 0.16}$	$1.45_{\pm 0.03}$	$1.07_{\pm 0.11}$	$1.01_{\pm 0.09}$	$0.91_{\pm 0.03}$	$2.14_{\pm 1.68}$
	$\text{TB}_{\theta,\varphi}$	(learnt var.)	$1.77_{\pm 0.17}$	$1.76_{\pm 0.27}$	$1.07_{\pm 0.15}$	$1.13_{\pm 0.11}$	$1.05_{\pm 0.20}$	$1.17_{\pm 0.24}$
	PIS_θ	(fixed var.)	$5.71_{\pm 0.10}$	$5.43_{\pm 0.07}$	$5.22_{\pm 0.04}$	$5.24_{\pm 0.08}$	$5.63_{\pm 0.29}$	$5.31_{\pm 0.36}$
	PIS_θ	(learnt var.)	$5.90_{\pm 1.46}$	$4.68_{\pm 0.08}$	$4.56_{\pm 0.09}$	$4.46_{\pm 0.06}$	$4.58_{\pm 0.08}$	$4.87_{\pm 0.26}$
	PIS_θ + TLM_φ	(learnt var.)	$4.09_{\pm 0.53}$	$3.47_{\pm 0.26}$	$3.75_{\pm 0.38}$	$4.39_{\pm 0.04}$	$4.37_{\pm 0.11}$	$6.10_{\pm 1.24}$
	PIS_θ + VarGrad_φ	(learnt var.)	$6.98_{\pm 0.03}$	$5.80_{\pm 1.08}$	$4.35_{\pm 0.09}$	$4.17_{\pm 0.27}$	$4.38_{\pm 0.05}$	$4.40_{\pm 0.11}$
	Ground Truth				$1.10_{\pm 0.14}$			

Table 6: Comparison of 4 algorithms by ELBO, EUBO, and 2-Wasserstein between generated and ground truth samples with varying number of discretisation steps on 40GMM. Mean and std over 3 runs are specified.

ELBO (↑)

Energy ↓	Method ↓ Steps →		$T=2$	$T=3$	$T=4$	$T=5$	$T=10$	$T=15$	$T=20$	$T=25$
40GMM	TB_θ	(fixed var.)	$-4.52_{\pm 1.06}$	$-12.95_{\pm 13.36}$	$-3.75_{\pm 0.81}$	$-3.60_{\pm 1.03}$	$-2.40_{\pm 0.06}$	$-2.30_{\pm 0.02}$	$-2.83_{\pm 0.82}$	$-2.29_{\pm 0.12}$
	TB_θ + TLM_φ	(fixed var.)	$-4.96_{\pm 0.78}$	$-3.11_{\pm 0.21}$	$-2.62_{\pm 0.09}$	$-2.46_{\pm 0.05}$	$-2.17_{\pm 0.01}$	$-2.04_{\pm 0.07}$	$-1.98_{\pm 0.04}$	$-1.96_{\pm 0.04}$
	$\text{TB}_{\theta,\varphi}$	(fixed var.)	$-4.18_{\pm 1.18}$	$-3.35_{\pm 0.02}$	$-2.81_{\pm 0.07}$	$-2.48_{\pm 0.06}$	$-2.26_{\pm 0.03}$	$-2.06_{\pm 0.08}$	$-1.95_{\pm 0.09}$	$-2.00_{\pm 0.08}$
	TB_θ	(learnt var.)	$-5.30_{\pm 0.65}$	$-3.96_{\pm 0.92}$	$-4.96_{\pm 1.15}$	$-3.58_{\pm 0.04}$	$-2.88_{\pm 0.64}$	$-2.26_{\pm 0.13}$	$-2.08_{\pm 0.14}$	$-1.94_{\pm 0.04}$
	TB_θ + TLM_φ	(learnt var.)	$-3.76_{\pm 0.34}$	$-2.64_{\pm 0.12}$	$-2.03_{\pm 0.06}$	$-1.74_{\pm 0.03}$	$-1.28_{\pm 0.03}$	$-1.14_{\pm 0.13}$	$-1.25_{\pm 0.19}$	$-1.34_{\pm 0.15}$
	$\text{TB}_{\theta,\varphi}$	(learnt var.)	$-4.12_{\pm 0.31}$	$-2.60_{\pm 0.12}$	$-2.44_{\pm 0.17}$	$-1.90_{\pm 0.03}$	$-1.26_{\pm 0.11}$	$-1.15_{\pm 0.10}$	$-0.99_{\pm 0.09}$	$-1.03_{\pm 0.21}$
	PIS_θ	(fixed var.)	$-4.45_{\pm 0.03}$	$-3.67_{\pm 0.70}$	$-3.56_{\pm 0.66}$	$-3.95_{\pm 0.00}$	$-3.83_{\pm 0.01}$	$-3.33_{\pm 0.64}$	$-3.31_{\pm 0.64}$	$-3.76_{\pm 0.00}$
	PIS_θ	(learnt var.)	$-3.34_{\pm 0.49}$	$-2.93_{\pm 0.53}$	$-2.90_{\pm 0.55}$	$-2.87_{\pm 0.58}$	$-2.38_{\pm 0.01}$	$-3.20_{\pm 0.69}$	$-2.77_{\pm 0.65}$	$-2.29_{\pm 0.08}$
	PIS_θ + TLM_φ	(learnt var.)	$-2.53_{\pm 0.11}$	$-2.62_{\pm 0.00}$	$-2.22_{\pm 0.08}$	$-2.34_{\pm 0.02}$	$-2.29_{\pm 0.05}$	$-2.33_{\pm 0.01}$	$-2.32_{\pm 0.01}$	$-2.32_{\pm 0.01}$
	PIS_θ + VarGrad_φ	(learnt var.)	$-2.69_{\pm 0.04}$	$-2.55_{\pm 0.13}$	$-2.47_{\pm 0.06}$	$-2.79_{\pm 0.64}$	$-2.73_{\pm 0.69}$	$-2.72_{\pm 0.69}$	$-2.27_{\pm 0.09}$	$-2.26_{\pm 0.09}$

EUBO (↓)

Energy ↓	Method ↓ Steps →		$T=2$	$T=3$	$T=4$	$T=5$	$T=10$	$T=15$	$T=20$	$T=25$
40GMM	TB_θ	(fixed var.)	$141.80_{\pm 22.85}$	$97.40_{\pm 4.91}$	$251.06_{\pm 95.60}$	$102.98_{\pm 29.73}$	$102.84_{\pm 10.99}$	$2732.36_{\pm 3753.31}$	$121.65_{\pm 26.31}$	$98.32_{\pm 22.12}$
	TB_θ + TLM_φ	(fixed var.)	$76.05_{\pm 2.25}$	$70.30_{\pm 14.86}$	$98.52_{\pm 7.47}$	$82.80_{\pm 8.02}$	$117.96_{\pm 8.84}$	$114.55_{\pm 12.78}$	$108.29_{\pm 18.38}$	$84.96_{\pm 23.71}$
	$\text{TB}_{\theta,\varphi}$	(fixed var.)	$87.03_{\pm 26.30}$	$58.65_{\pm 2.54}$	$97.50_{\pm 17.17}$	$104.46_{\pm 15.08}$	$116.41_{\pm 12.13}$	$116.25_{\pm 34.53}$	$115.79_{\pm 24.49}$	$107.80_{\pm 6.42}$
	TB_θ	(learnt var.)	$5.33_{\pm 0.15}$	$7.86_{\pm 2.26}$	$4.44_{\pm 0.77}$	$38.06_{\pm 47.29}$	$5.98_{\pm 2.29}$	$6.89_{\pm 4.76}$	$5.29_{\pm 3.08}$	$37.75_{\pm 46.22}$
	TB_θ + TLM_φ	(learnt var.)	$2.74_{\pm 0.19}$	$3.38_{\pm 0.18}$	$3.88_{\pm 0.25}$	$6.11_{\pm 1.44}$	$7.25_{\pm 0.88}$	$5.84_{\pm 0.50}$	$8.26_{\pm 2.03}$	$14.95_{\pm 10.41}$
	$\text{TB}_{\theta,\varphi}$	(learnt var.)	$3.17_{\pm 0.14}$	$3.20_{\pm 0.52}$	$3.18_{\pm 0.29}$	$5.34_{\pm 0.66}$	$4.56_{\pm 0.22}$	$4.64_{\pm 0.17}$	$4.70_{\pm 0.36}$	$5.90_{\pm 1.03}$
	PIS_θ	(fixed var.)	$108.22_{\pm 0.38}$	$106.93_{\pm 1.45}$	$106.20_{\pm 2.48}$	$108.22_{\pm 0.38}$	$108.22_{\pm 0.38}$	$108.22_{\pm 0.38}$	$106.84_{\pm 1.57}$	$106.39_{\pm 2.21}$
	PIS_θ	(learnt var.)	$106.96_{\pm 0.43}$	$106.97_{\pm 0.45}$	$104.85_{\pm 2.55}$	$106.97_{\pm 0.46}$	$106.99_{\pm 0.48}$	$104.08_{\pm 3.64}$	$104.94_{\pm 3.17}$	$99.36_{\pm 5.90}$
	PIS_θ + TLM_φ	(learnt var.)	$106.95_{\pm 0.42}$	$106.95_{\pm 0.44}$	$106.95_{\pm 0.42}$	$106.99_{\pm 0.37}$	$106.98_{\pm 0.40}$	$107.00_{\pm 0.40}$	$99.51_{\pm 11.05}$	$105.41_{\pm 2.71}$
	PIS_θ + VarGrad_φ	(learnt var.)	$106.95_{\pm 0.42}$	$106.95_{\pm 0.44}$	$106.95_{\pm 0.42}$	$106.99_{\pm 0.37}$	$106.98_{\pm 0.40}$	$102.54_{\pm 5.94}$	$106.51_{\pm 1.12}$	$104.24_{\pm 4.27}$

2-Wasserstein (↓)

Energy ↓	Method ↓ Steps →		$T=2$	$T=3$	$T=4$	$T=5$	$T=10$	$T=15$	$T=20$	$T=25$
40GMM	TB_θ	(fixed var.)	$38.72_{\pm 5.09}$	$33.46_{\pm 2.17}$	$33.93_{\pm 2.24}$	$31.12_{\pm 2.34}$	$29.18_{\pm 1.39}$	$29.93_{\pm 0.16}$	$27.91_{\pm 2.44}$	$29.97_{\pm 0.63}$
	TB_θ + TLM_φ	(fixed var.)	$34.00_{\pm 2.88}$	$29.13_{\pm 0.14}$	$29.27_{\pm 0.15}$	$29.19_{\pm 0.08}$	$28.28_{\pm 0.13}$	$27.45_{\pm 0.36}$	$25.74_{\pm 1.76}$	$24.94_{\pm 2.66}$
	$\text{TB}_{\theta,\varphi}$	(fixed var.)	$31.65_{\pm 2.70}$	$29.21_{\pm 0.09}$	$28.97_{\pm 0.27}$	$28.42_{\pm 0.57}$	$27.00_{\pm 0.53}$	$26.84_{\pm 0.65}$	$26.21_{\pm 1.33}$	$25.54_{\pm 1.74}$
	TB_θ	(learnt var.)	$18.36_{\pm 0.71}$	$20.59_{\pm 2.00}$	$16.34_{\pm 1.75}$	$21.99_{\pm 6.57}$	$15.63_{\pm 2.28}$	$15.14_{\pm 1.33}$	$14.02_{\pm 0.28}$	$19.38_{\pm 7.77}$
	TB_θ + TLM_φ	(learnt var.)	$10.02_{\pm 0.23}$	$12.10_{\pm 2.73}$	$15.53_{\pm 0.71}$	$19.84_{\pm 1.53}$	$18.85_{\pm 0.74}$	$18.08_{\pm 1.74}$	$19.64_{\pm 1.49}$	$20.98_{\pm 1.49}$
	$\text{TB}_{\theta,\varphi}$	(learnt var.)	$10.79_{\pm 0.39}$	$11.47_{\pm 1.38}$	$11.29_{\pm 1.60}$	$19.37_{\pm 1.51}$	$16.45_{\pm 0.79}$	$16.18_{\pm 0.61}$	$16.82_{\pm 1.22}$	$17.83_{\pm 1.77}$
	PIS_θ	(fixed var.)	$30.44_{\pm 0.07}$	$30.45_{\pm 0.05}$	$30.44_{\pm 0.08}$	$30.43_{\pm 0.07}$	$30.46_{\pm 0.04}$	$30.47_{\pm 0.09}$	$30.46_{\pm 0.05}$	$30.46_{\pm 0.06}$
	PIS_θ	(learnt var.)	$30.43_{\pm 0.07}$	$30.37_{\pm 0.14}$	$30.15_{\pm 0.18}$	$30.09_{\pm 0.28}$	$30.09_{\pm 0.21}$	$30.15_{\pm 0.35}$	$30.14_{\pm 0.45}$	$29.73_{\pm 0.60}$
	PIS_θ + TLM_φ	(learnt var.)	$30.22_{\pm 0.35}$	$30.14_{\pm 0.21}$	$29.93_{\pm 0.57}$	$29.92_{\pm 0.07}$	$29.63_{\pm 0.27}$	$29.91_{\pm 0.26}$	$29.85_{\pm 0.17}$	$29.86_{\pm 0.34}$
	PIS_θ + VarGrad_φ	(learnt var.)	$28.33_{\pm 3.00}$	$30.42_{\pm 0.08}$	$30.08_{\pm 0.25}$	$30.06_{\pm 0.50}$	$30.46_{\pm 0.06}$	$30.15_{\pm 0.49}$	$30.22_{\pm 0.24}$	$30.16_{\pm 0.39}$
	Ground Truth					$3.95_{\pm 0.51}$				

Table 7: Comparison of 4 algorithms by ELBO, EUBO, and 2-Wasserstein between generated and ground truth samples with varying number of discretisation steps on Easy Funnel and Hard Funnel. Mean and std over 3 runs are specified.

ELBO (↑)

Energy ↓	Method ↓ Steps →		$T = 5$	$T = 10$	$T = 15$	$T = 20$	$T = 25$	$T = 30$
Easy Funnel	TB_θ	(fixed var.)	$-0.61_{\pm 0.00}$	$-0.34_{\pm 0.02}$	$-0.23_{\pm 0.02}$	$-0.19_{\pm 0.01}$	$-0.15_{\pm 0.00}$	$-0.14_{\pm 0.00}$
	$\text{TB}_\theta + \text{TLM}_\varphi$	(fixed var.)	$-0.58_{\pm 0.02}$	$-0.31_{\pm 0.01}$	diverged	diverged	diverged	diverged
	$\text{TB}_{\theta,\varphi}$	(fixed var.)	$-0.62_{\pm 0.03}$	$-0.33_{\pm 0.01}$	$-0.22_{\pm 0.01}$	$-0.17_{\pm 0.00}$	$-0.16_{\pm 0.01}$	$-0.13_{\pm 0.01}$
	TB_θ	(learnt var.)	$-0.34_{\pm 0.01}$	$-0.17_{\pm 0.01}$	$-0.11_{\pm 0.00}$	$-0.08_{\pm 0.01}$	$-0.07_{\pm 0.01}$	$-0.07_{\pm 0.01}$
	$\text{TB}_\theta + \text{TLM}_\varphi$	(learnt var.)	$-0.08_{\pm 0.04}$	$-0.11_{\pm 0.01}$	diverged	diverged	diverged	diverged
	$\text{TB}_{\theta,\varphi}$	(learnt var.)	$-0.15_{\pm 0.01}$	$-0.11_{\pm 0.01}$	$-0.08_{\pm 0.00}$	$-0.06_{\pm 0.01}$	$-0.05_{\pm 0.00}$	$-0.05_{\pm 0.00}$
Hard Funnel	TB_θ	(fixed var.)	$-1.68_{\pm 0.02}$	$-1.17_{\pm 0.03}$	$-0.98_{\pm 0.01}$	$-0.89_{\pm 0.02}$	$-0.79_{\pm 0.00}$	$-0.76_{\pm 0.01}$
	$\text{TB}_\theta + \text{TLM}_\varphi$	(fixed var.)	$-1.43_{\pm 0.01}$	$-1.09_{\pm 0.03}$	diverged	diverged	diverged	diverged
	$\text{TB}_{\theta,\varphi}$	(fixed var.)	$-1.83_{\pm 0.08}$	$-1.58_{\pm 0.45}$	$-0.95_{\pm 0.01}$	$-0.85_{\pm 0.02}$	$-0.79_{\pm 0.03}$	$-0.76_{\pm 0.01}$
	TB_θ	(learnt var.)	$-1.72_{\pm 0.04}$	$-1.25_{\pm 0.03}$	$-0.91_{\pm 0.05}$	$-0.61_{\pm 0.01}$	$-0.48_{\pm 0.02}$	$-0.44_{\pm 0.01}$
	$\text{TB}_\theta + \text{TLM}_\varphi$	(learnt var.)	$-0.63_{\pm 0.26}$	$-0.45_{\pm 0.04}$	diverged	diverged	diverged	diverged
	$\text{TB}_{\theta,\varphi}$	(learnt var.)	$-1.01_{\pm 0.04}$	$-0.69_{\pm 0.01}$	$-0.56_{\pm 0.03}$	$-0.55_{\pm 0.08}$	$-0.41_{\pm 0.02}$	$-0.41_{\pm 0.03}$

EUBO (↓)

Energy ↓	Method ↓ Steps →		$T = 5$	$T = 10$	$T = 15$	$T = 20$	$T = 25$	$T = 30$
Easy Funnel	TB_θ	(fixed var.)	$0.73_{\pm 0.01}$	$0.40_{\pm 0.02}$	$0.28_{\pm 0.01}$	$0.21_{\pm 0.01}$	$0.17_{\pm 0.00}$	$0.15_{\pm 0.00}$
	$\text{TB}_\theta + \text{TLM}_\varphi$	(fixed var.)	$0.64_{\pm 0.02}$	$0.35_{\pm 0.01}$	$2.80_{\pm 0.11}$	diverged	diverged	diverged
	$\text{TB}_{\theta,\varphi}$	(fixed var.)	$0.84_{\pm 0.03}$	$0.39_{\pm 0.04}$	$0.26_{\pm 0.01}$	$0.20_{\pm 0.01}$	$0.18_{\pm 0.01}$	$0.15_{\pm 0.02}$
	TB_θ	(learnt var.)	$0.38_{\pm 0.02}$	$0.18_{\pm 0.01}$	$0.12_{\pm 0.00}$	$0.09_{\pm 0.00}$	$0.08_{\pm 0.00}$	$0.06_{\pm 0.00}$
	$\text{TB}_\theta + \text{TLM}_\varphi$	(learnt var.)	$0.08_{\pm 0.03}$	$0.12_{\pm 0.01}$	diverged	diverged	diverged	diverged
	$\text{TB}_{\theta,\varphi}$	(learnt var.)	$0.17_{\pm 0.03}$	$0.12_{\pm 0.01}$	$0.08_{\pm 0.00}$	$0.07_{\pm 0.01}$	$0.05_{\pm 0.00}$	$0.05_{\pm 0.00}$
Hard Funnel	TB_θ	(fixed var.)	$95.36_{\pm 8.41}$	$78.77_{\pm 8.35}$	$75.40_{\pm 3.35}$	$72.83_{\pm 5.15}$	$70.99_{\pm 4.26}$	$68.93_{\pm 5.01}$
	$\text{TB}_\theta + \text{TLM}_\varphi$	(fixed var.)	$109.52_{\pm 52.22}$	$55.76_{\pm 17.37}$	$285.59_{\pm 53.98}$	diverged	diverged	diverged
	$\text{TB}_{\theta,\varphi}$	(fixed var.)	$374.26_{\pm 228.11}$	$694.95_{\pm 901.63}$	$79.03_{\pm 50.28}$	$69.47_{\pm 30.67}$	$56.34_{\pm 26.13}$	$95.48_{\pm 24.76}$
	TB_θ	(learnt var.)	$155.51_{\pm 19.60}$	$102.97_{\pm 41.86}$	$105.67_{\pm 6.26}$	$121.76_{\pm 41.45}$	$349.17_{\pm 406.05}$	$79.37_{\pm 33.30}$
	$\text{TB}_\theta + \text{TLM}_\varphi$	(learnt var.)	$1509.44_{\pm 2037.59}$	$22.77_{\pm 14.77}$	diverged	diverged	diverged	diverged
	$\text{TB}_{\theta,\varphi}$	(learnt var.)	$2800.63_{\pm 1351.92}$	$21.48_{\pm 13.09}$	$62.95_{\pm 75.95}$	$30.35_{\pm 20.62}$	$8.60_{\pm 0.97}$	$17.29_{\pm 8.63}$

2-Wasserstein (↓)

Energy ↓	Method ↓ Steps →		$T = 5$	$T = 10$	$T = 15$	$T = 20$	$T = 25$	$T = 30$
Easy Funnel	TB_θ	(fixed var.)	$2.45_{\pm 0.04}$	$2.43_{\pm 0.03}$	$2.45_{\pm 0.04}$	$2.49_{\pm 0.01}$	$2.47_{\pm 0.02}$	$2.48_{\pm 0.03}$
	$\text{TB}_\theta + \text{TLM}_\varphi$	(fixed var.)	$2.44_{\pm 0.02}$	$2.45_{\pm 0.03}$	$2.60_{\pm 0.03}$	diverged	diverged	diverged
	$\text{TB}_{\theta,\varphi}$	(fixed var.)	$2.45_{\pm 0.04}$	$2.45_{\pm 0.01}$	$2.45_{\pm 0.01}$	$2.47_{\pm 0.02}$	$2.49_{\pm 0.03}$	$2.49_{\pm 0.03}$
	TB_θ	(learnt var.)	$2.49_{\pm 0.02}$	$2.50_{\pm 0.03}$	$2.49_{\pm 0.03}$	$2.49_{\pm 0.02}$	$2.50_{\pm 0.02}$	$2.53_{\pm 0.01}$
	$\text{TB}_\theta + \text{TLM}_\varphi$	(learnt var.)	$2.51_{\pm 0.05}$	$2.53_{\pm 0.02}$	diverged	diverged	diverged	diverged
	$\text{TB}_{\theta,\varphi}$	(learnt var.)	$2.52_{\pm 0.01}$	$2.49_{\pm 0.02}$	$2.51_{\pm 0.03}$	$2.53_{\pm 0.02}$	$2.52_{\pm 0.02}$	$2.50_{\pm 0.02}$
	Ground Truth				$2.58_{\pm 0.05}$			
Hard Funnel	TB_θ	(fixed var.)	$22.82_{\pm 0.73}$	$22.50_{\pm 0.75}$	$22.36_{\pm 0.78}$	$22.17_{\pm 0.81}$	$22.08_{\pm 0.79}$	$21.99_{\pm 0.79}$
	$\text{TB}_\theta + \text{TLM}_\varphi$	(fixed var.)	$22.43_{\pm 2.39}$	$22.05_{\pm 2.41}$	$23.07_{\pm 2.36}$	diverged	diverged	diverged
	$\text{TB}_{\theta,\varphi}$	(fixed var.)	$22.53_{\pm 2.39}$	$22.47_{\pm 2.50}$	$21.90_{\pm 2.40}$	$21.53_{\pm 2.53}$	$21.26_{\pm 2.48}$	$21.18_{\pm 2.52}$
	TB_θ	(learnt var.)	$22.45_{\pm 0.79}$	$21.94_{\pm 0.85}$	$21.63_{\pm 0.80}$	$21.53_{\pm 0.76}$	$21.44_{\pm 0.81}$	$21.36_{\pm 0.81}$
	$\text{TB}_\theta + \text{TLM}_\varphi$	(learnt var.)	$21.32_{\pm 0.90}$	$21.04_{\pm 0.84}$	diverged	diverged	diverged	diverged
	$\text{TB}_{\theta,\varphi}$	(learnt var.)	$21.48_{\pm 0.81}$	$20.96_{\pm 0.88}$	$21.04_{\pm 0.91}$	$20.91_{\pm 0.86}$	$21.03_{\pm 0.80}$	$21.01_{\pm 0.90}$
	Ground Truth				$24.24_{\pm 4.20}$			

Table 8: Comparison of 4 algorithms by ELBO gap, EUBO gap, and 2-Wasserstein between generated and ground truth samples with varying number of discretisation steps on Manywell and Distorted Manywell. Mean and std over 3 runs are specified.

ELBO gap (↑)

Energy ↓	Method ↓ Steps →	$T = 5$	$T = 10$	$T = 20$
ManyWell	TB_θ (fixed var.)	$-36.25_{\pm 13.68}$	$-12.36_{\pm 0.10}$	$-5.67_{\pm 0.05}$
	TB_θ (learnt var.)	$-2.40_{\pm 0.03}$	$-0.78_{\pm 0.02}$	$-0.41_{\pm 0.09}$
	TB_θ + TLM_φ	$-2.16_{\pm 0.12}$	$-0.79_{\pm 0.01}$	diverged
	$\text{TB}_{\theta,\varphi}$	$-2.22_{\pm 0.05}$	$-0.83_{\pm 0.02}$	$-0.32_{\pm 0.01}$
Distorted ManyWell	TB_θ (fixed var.)	$-77.74_{\pm 6.27}$	$-25.75_{\pm 9.45}$	$-11.96_{\pm 0.16}$
	TB_θ (learnt var.)	$-8.36_{\pm 0.73}$	$-5.70_{\pm 1.04}$	$-3.83_{\pm 0.13}$
	TB_θ + TLM_φ	$-6.47_{\pm 1.12}$	$-3.99_{\pm 2.31}$	diverged
	$\text{TB}_{\theta,\varphi}$	$-7.33_{\pm 1.34}$	$-4.07_{\pm 2.38}$	$-2.90_{\pm 1.60}$

EUBO gap (↓)

Energy ↓	Method ↓ Steps →	$T = 5$	$T = 10$	$T = 20$
ManyWell	TB_θ (fixed var.)	$12.80_{\pm 3.02}$	$6.72_{\pm 0.02}$	$4.06_{\pm 0.04}$
	TB_θ (learnt var.)	$1.68_{\pm 0.02}$	$0.63_{\pm 0.01}$	$0.36_{\pm 0.09}$
	TB_θ + TLM_φ	$1.69_{\pm 0.01}$	$0.72_{\pm 0.02}$	diverged
	$\text{TB}_{\theta,\varphi}$	$1.61_{\pm 0.01}$	$0.64_{\pm 0.01}$	$0.31_{\pm 0.01}$
Distorted ManyWell	TB_θ (fixed var.)	$19.68_{\pm 0.62}$	$10.46_{\pm 2.30}$	$6.19_{\pm 0.10}$
	TB_θ (learnt var.)	$4.53_{\pm 0.30}$	$3.16_{\pm 0.37}$	$2.15_{\pm 0.08}$
	TB_θ + TLM_φ	$4.09_{\pm 0.49}$	$2.39_{\pm 1.09}$	diverged
	$\text{TB}_{\theta,\varphi}$	$4.29_{\pm 0.42}$	$2.38_{\pm 1.19}$	$1.77_{\pm 0.80}$

2-Wasserstein (↓)

Energy ↓	Method ↓ Steps →	$T = 5$	$T = 10$	$T = 20$
ManyWell	TB_θ (fixed var.)	$5.57_{\pm 0.11}$	$5.40_{\pm 0.02}$	$5.38_{\pm 0.01}$
	TB_θ (learnt var.)	$5.35_{\pm 0.01}$	$5.38_{\pm 0.01}$	$5.41_{\pm 0.03}$
	TB_θ + TLM_φ	$5.33_{\pm 0.03}$	$5.36_{\pm 0.02}$	diverged
	$\text{TB}_{\theta,\varphi}$	$5.36_{\pm 0.02}$	$5.39_{\pm 0.01}$	$5.40_{\pm 0.01}$
	Ground Truth		$5.42_{\pm 0.02}$	
Distorted ManyWell	TB_θ (fixed var.)	$5.87_{\pm 0.03}$	$5.66_{\pm 0.02}$	$5.53_{\pm 0.06}$
	TB_θ (learnt var.)	$5.53_{\pm 0.00}$	$5.44_{\pm 0.03}$	$5.40_{\pm 0.01}$
	TB_θ + TLM_φ	$5.50_{\pm 0.02}$	$5.42_{\pm 0.09}$	diverged
	$\text{TB}_{\theta,\varphi}$	$5.54_{\pm 0.01}$	$5.41_{\pm 0.08}$	$5.37_{\pm 0.06}$
	Ground Truth		$5.30_{\pm 0.01}$	

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We introduce the main topic of our work — adaptive destruction processes for diffusion samplers — in the abstract and introduction, which is the main focus of the whole paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed where relevant throughout the text, such as instability of some methods considered (Appendix D.1), unsuccessful techniques (Appendix C.1), and the introduction of new hyperparameters that need to be tuned.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We have no (significant) theoretical results, but Appendix B.1 contains a derivation that holds under assumptions from [7], as mentioned in the main text, and Appendix B.4 extends a known connection between sampling and soft RL to the diffusion samplers case.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: See §3, Appendix E.3 and Appendix E.4.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [NA]

   Justification: The workshop does not allow for uploading of additional supplementary materials.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See §3, Appendix E.3 and Appendix E.4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All results on synthetic targets are reported taking statistical significance into account, e.g., see Table 3 and other tables in Appendix F. We do not report error bars on the GAN experiments due to computation constraints.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialisation, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix E.3 and Appendix E.4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: To the best of our knowledge, we follow the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The authors do not believe that this question is applicable to the main problem studied in this paper (diffusion sampling). Improved sampling algorithms have downstream applications, but we do not feel there is any specific impact of our work that must be discussed.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The authors believe this question is not applicable to the topic of the paper.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All code and data we use is free to use in academic research and we credit it where required.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: Full code for reproducing experiments will be released with the publication of the paper. A detailed explanation of the training setup can be found in Appendix E.3

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Crowdsourcing and research with human subjects were not used for this paper.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This question is not applicable to the content of this paper.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The topic of this paper is not directly relevant to LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.