VarDiU: A Variational Diffusive Upper Bound for One-Step Diffusion Distillation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recently, diffusion distillation methods have compressed thousand-step teacher diffusion models into one-step student generators while preserving sample quality. Most existing approaches train the student model using a diffusive divergence whose gradient is approximated via the student's score function, learned through denoising score matching (DSM). Since DSM training is imperfect, the resulting gradient estimate is inevitably biased, leading to sub-optimal performance. In this paper, we propose *VarDiU* (pronounced /va:rdju:/), a Variational Diffusive Upper Bound that admits an unbiased gradient estimator and can be directly applied to diffusion distillation. Using this objective, we compare our method with Diff-Instruct and demonstrate that it achieves higher generation quality and enables a more efficient and stable training procedure for one-step diffusion distillation.

1 Introduction

Diffusion models [38, 15, 42, 44] have achieved remarkable success in generating high-quality samples. However, sampling typically requires a large number of denoising steps, making the process computationally inefficient in practice. To address this limitation, many acceleration methods [15, 31, 39, 23, 25, 2, 1, 32, 5, 50, 54] have been proposed to reduce the number of function evaluations (NFEs) in pre-trained diffusion models, lowering the cost from thousands of steps to only tens. More recently, distillation-based approaches have further compressed the sampling process to as few as a single step while largely preserving generation quality [61, 36, 4, 41, 14, 18, 21, 28, 37, 51, 57].

In this paper, we investigate a family of distillation methods based on divergence minimisation [28, 60, 57, 61]. These methods compress the full denoising process of a pre-trained teacher diffusion model into a one-step latent variable model by minimising a diffusive divergence (e.g. Diffusive Reverse KL Divergence [13, 57, 28]) between the student and teacher through their respective score estimations. Although these approaches have demonstrated promising results, their gradients are approximated using the student's score function, which is learned through denoising score matching (DSM). Because DSM training is inherently imperfect, the resulting gradient estimates are inevitably biased, often leading to sub-optimal generation performance.

In this study, we introduce *Variational Diffusive Upper Bound Distillation (VarDiU)*, a method that formulates a variational upper bound of the diffusive divergence whose gradient can be estimated **unbiasedly**. We demonstrate that this unbiased gradient estimation leads to more stable and efficient training, achieving better performance compared to the original diffusive divergence, which relies on biased gradient approximations.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

2 Background on Diffusion Models

Let $p_d(x_0)$ denote the data distribution. Diffusion models define a forward noising process as $q(x_{0:T}) = p_d(x_0)\prod_{t=1}^{T} q(x_t|x_{t-1})$ where the transition kernel is a fixed Gaussian distribution $q(x_t|x_{t-1}) = \mathcal{N}(x_t|\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$. This process admits the closed-form distribution,

$$q(x_t|x_0) = \mathcal{N}(x_t|\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I), \tag{1}$$

with $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$. As $T \to \infty$, the final state x_T converges to a standard Gaussian distribution, i.e., $q(x_T) \to \mathcal{N}(0, I)$. The generative process also uses a sequence of Gaussian distributions:

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}|\mu_{t-1}(x_t), \Sigma_{t-1}(x_t))\,, \tag{2}$$

where the covariance $\Sigma_{t-1}(x_t)$ may either be learned [31, 32, 1] or fixed [2, 15]. The mean is estimated via Tweedie's formula [10, 34]:

$$\mu_{t-1}(x_t) = \frac{1}{\sqrt{1-\beta_t}}\Big(x_t + \beta_t \nabla_{x_t}\log p_d(x_t)\Big), \tag{3}$$

where the score function $\nabla_{x_t}\log p_d(x_t)$ is approximated with a network $\boldsymbol{s}_\psi(x_t; t)$ which is learned through denoising score matching (DSM) [48, 42]:

$$\mathcal{L}_{\mathrm{DSM}}(\psi) = \int p_d(x_0)\, q(x_t|x_0)\, \big\|\boldsymbol{s}_\psi(x_t;t) - \nabla_{x_t}\log q(x_t|x_0)\big\|_2^2\, \mathrm{d}x_t\, \mathrm{d}x_0. \tag{4}$$

The classical diffusion models require thousands of time steps to generate high-quality samples, making them computationally expensive. In the following section, we discuss a family of methods that can distill multi-step diffusion processes into one-step generative models.

2.1 Score-Based Distillation via Diffusive Reverse KL Minimisation

Score-based distillation methods aim to compress a teacher diffusion model, which we also denote as p_d, into a one-step implicit generative model [11, 16, 58], defined as

$$p_\theta(x_0) = \int \delta(x_0 - g_\theta(z))\, p(z)\, dz, \tag{5}$$

where $\delta(\cdot)$ is the Dirac delta function, $p(z)$ is a standard Gaussian prior, and g_θ is a deterministic neural network that generates samples in a single step. The training objective is formulated as the *Diffusive KL divergence* (DiKL) [58, 49, 28, 13, 57], which measures the discrepancy between the student and teacher distributions across all intermediate noise levels:

$$\mathrm{DiKL}(p_\theta||p_d) \equiv \int_0^1 \omega(t)\, \mathrm{KL}(p_\theta^{(t)}(x_t)\|p_d^{(t)}(x_t))\, \mathrm{d}t, \tag{6}$$

where $\omega(t)$ is a positive weighting function. Here, $p_\theta^{(t)}(x_t) = \int q(x_t|x_0)\, p_\theta(x_0)\, dx_0$, and $p_d^{(t)}(x_t)$ is the marginal distribution induced by the pre-trained teacher diffusion model under the same noise kernel $q(x_t|x_0)$. This divergence is well defined even when the underlying distributions have disjoint support or don't even allow density functions [58]. However, direct estimation of this divergence is intractable. One can derive the gradient w.r.t. θ at time t as

$$\nabla_\theta \mathrm{KL}(p_\theta^{(t)}\|p_d^{(t)}) = \int \left[\Big(\nabla_{x_t}\log p_\theta^{(t)}(x_t) - \nabla_{x_t}\log p_d^{(t)}(x_t)\Big)\frac{\partial x_t}{\partial \theta}\right] \mathrm{d}x_t, \tag{7}$$

see A.2 in [28] for a derivation. The teacher score $\nabla_{x_t}\log p_d^{(t)}(x_t)$ is available from the pre-trained diffusion model. However, the student score $\nabla_{x_t}\log p_\theta^{(t)}(x_t)$ is unknown. Fortunately, since we can sample from the student distribution, it can be estimated using denoising score matching (DSM):

$$\mathcal{L}_{\mathrm{DSM}}(\psi') = \int p_\theta(x_0)\, q(x_t|x_0)\, \big\|\boldsymbol{s}_{\psi'}(x_t;t) - \nabla_{x_t}\log q(x_t|x_0)\big\|_2^2\, \mathrm{d}x_t\, \mathrm{d}x_0. \tag{8}$$

This method is known as VSD [49] or Diff-Instruct [28] in the context of diffusion distillation. In practice, DSM approximations are imperfect due to the limited capacity of the score network $\boldsymbol{s}_{\psi'}$. As a result, *the direct gradient estimation of DiKL is biased*, which leads to suboptimal training of the distilled student model. In the next section, we introduce a variational upper bound on the reverse KL divergence that enables unbiased gradient estimation for diffusion distillation.

3 Variational Diffusive Upper Bound Distillation

Instead of relying on optimisation-based estimation of the DiKL objective, we propose a variational upper bound whose gradient can be estimated unbiasedly. Specifically, we propose to extend the RKL upper bound of [56] into the diffusion space,

$$\mathrm{KL}(p_\theta^{(t)}(x)||p_d^{(t)}(x)) \leq \mathrm{KL}(p_\theta^{(t)}(x_t|z)p(z)||p_d^{(t)}(x)q_\phi^{(t)}(z|x_t)) \equiv \mathrm{U}^{(t)}(\theta,\phi).$$

The inequality follows by Jensen's inequality; and the equality can be attained when the variational posterior is equal to the true posterior $q_\phi(z|x_t;t) = p_\theta(z|x_t) \propto p_\theta(x_t|z)p(z)$ (see Appendix A for a proof). We are then ready to define the diffusive upper bound as

$$\mathrm{DiU}(\theta,\phi) \equiv \int_0^1 w(t)\mathrm{U}^{(t)}(\theta,\phi)\mathrm{d}t \geq \int_0^1 w(t)\mathrm{KL}(p_\theta^{(t)}||p_d^{(t)})\mathrm{d}t \equiv \mathrm{DiKL}(p_\theta||p_d), \tag{9}$$

where the bound is tight when $q_\phi^{(t)}(z|x_t) = p_\theta^{(t)}(z|x_t) \propto p_\theta^{(t)}(x_t|z)p(z)$ for all t almost everywhere.

3.1 Tractable Upper Bound Estimation

We then discuss how to estimate this upper bound. Writing the model joint as $p_\theta^{(t)}(x_t,z) = p_\theta^{(t)}(x_t|z)p(z)$, the upper bound at time step t can be written as

$$\mathrm{U}^{(t)}(\theta,\phi) = -\mathbb{H}(p_\theta^{(t)}(x_t,z)) - \iint p_\theta^{(t)}(x_t,z)\left(\log p_d^{(t)}(x_t)\mathrm{d}x_t - \log q_\phi^{(t)}(z|x_t)\right)\mathrm{d}z\mathrm{d}x_t. \tag{10}$$

where $\mathbb{H}(p_\theta^{(t)}(x_t,z))$ denotes the joint entropy, i.e. $\mathbb{H}(p) = -\int p(x)\log p(x)\mathrm{d}x$. In the original reverse KL upper bound in [56], the joint entropy is trained with the reparametrisation trick. In our case, when the student model is an implicit model, we present the following surprising fact that *the joint entropy is a constant*, which removes the requirement of joint entropy estimation.

Proposition 3.1. *Let $p_\theta(x,z) = p_\theta(x|z)p(z)$ with a fixed prior $p(z)$ and $p_\theta(x|z) = \mathcal{N}\big(x;\, g_\theta(z), \sigma^2 I\big)$. Then the joint entropy $\mathbb{H}(p_\theta(x,z))$ is a constant and independent of θ.*

See Appendix D for a proof. This should be distinguished from the marginal entropy $\mathbb{H}(p_\theta^{(t)}(x_t))$, which still depends on θ, also see Section 3.3 for a deeper discussion. This observation greatly simplifies training by eliminating the need for entropy estimation.

The second term of Equation 9 cannot be directly optimised via automatic differentiation because the teacher density $p_d^{(t)}(x_t)$ is unknown. However, in typical distillation settings, the score function $\nabla_{x_t}\log p_d^{(t)}(x_t)$ is assumed to be available. Leveraging this, we propose an alternative loss whose gradient is equivalent and can be computed via the score function:

$$\nabla_\theta \int p_\theta^{(t)}(x_t)\log p_d^{(t)}(x_t)\mathrm{d}x_t = \nabla_\theta \int p_\theta^{(t)}(x_t)\left(x_t^\top[\nabla_{x_t}\log p_d^{(t)}(x_t)]_{\mathrm{sg}}\right)\mathrm{d}x_t, \tag{11}$$

where $[\,\cdot\,]_{\mathrm{sg}}$ denotes the stop-gradient operator. The proof involves using the reparametrisation trick and chain rule; see Appendix B for details. The final loss objective at time t becomes

$$\mathrm{U}^{(t)}(\theta,\phi) \doteq -\iint p_\theta^{(t)}(x_t,z)\left(x_t^\top[\nabla_{x_t}\log p_d^{(t)}(x_t)]_{\mathrm{sg}} + \log q_\phi^{(t)}(z|x_t)\right)\mathrm{d}z\mathrm{d}x_t, \tag{12}$$

where $\doteq$ denotes equivalence up to a constant. Therefore, when using a simple Gaussian variational family $q_\phi^{(t)}(z|x_t)$, this bound can be unbiasedly estimated. A loss used practically can be found in Appendix C. We also propose a novel noise schedule and variance reduction techniques for training with this objective, see Appendix F.2 for details.

3.2 Improving Variational Inference with Normalising Flows

The upper bound becomes tight, i.e., equal to the DiKL objective, when the variational posterior matches the true posterior: $q_\phi^{(t)}(z|x_t) = p_\theta^{(t)}(z|x_t)$. Simple Gaussian variational distributions often

struggle to approximate complex posteriors, particularly in cases where the true posterior is multimodal or exhibits strong correlations. To address this limitation, we could use normalising flow [47, 46, 8, 33, 55, 9, 19] to increase the flexibility of the posterior [33]. Specifically, we define:

$$q_\phi^{(t)}(z|x_t) = \int \delta\left(z - f_\phi(a, x_t, t)\right) r_\phi(a|x_t; t)\, \mathrm{d}a, \tag{13}$$

where f is an invertible function and the base distribution is defined as a Gaussian indexed by t.

$$r_\phi(a|x_t; t) = \mathcal{N}\left(a; \mu_\phi(x_t; t),\, \sigma_t^2 \mathrm{diag}(\sigma_\phi^2(x_t; t))\right). \tag{14}$$

The time-dependent scaling factor σ_t modulates the entropy of the base distribution to match the noise level at diffusion step t. The corresponding log-density of $q_\phi^{(t)}(z|x_t)$ can be computed using the change-of-variables formula:

$$\log q_\phi^{(t)}(z = f_\phi(a; t)|x_t; t) = \log r_\phi(a|x_t; t) - \log \left| \det \frac{\partial f_\phi}{\partial a} \right|, \tag{15}$$

We can then substitute this log-density into Equation (12) to obtain a tractable objective for distillation.

3.3 An Information Maximisation View of the Variational Diffusive Upper Bound

In the previous section, we showed that the joint entropy term in Equation (12) is constant with respect to θ. As a result, the variational upper bound at diffusion step t simplifies to:

$$\mathrm{U}^{(t)}(\theta, \phi) = -\int p_\theta^{(t)}(x_t) \log p_d^{(t)}(x_t)\, \mathrm{d}x_t - \iint p_\theta^{(t)}(x_t, z) \log q_\phi^{(t)}(z|x_t)\, \mathrm{d}z\, \mathrm{d}x_t. \tag{16}$$

Now consider the original DiKL divergence at time step t, defined as:

$$\mathrm{KL}(p_\theta^{(t)} \,\|\, p_d^{(t)}) = -\int p_\theta^{(t)}(x_t) \log p_d^{(t)}(x_t)\, \mathrm{d}x_t + \int p_\theta^{(t)}(x_t) \log p_\theta^{(t)}(x_t)\, \mathrm{d}x_t. \tag{17}$$

We observe that the first term in both objectives (Equations (16) and (17)) is identical. The second term in Equation (17) corresponds to the negative entropy of the marginal distribution, i.e., $-\mathbb{H}(p_\theta^{(t)}(x_t))$, which depends on θ. We can then obtain an alternative perspective for justifying the proposed upper bound, formalised in the following theorem:

Theorem 3.2. *By the chain rule of entropy, we have:* $\mathbb{H}(p_\theta^{(t)}(x_t, z)) = \mathbb{H}(p_\theta^{(t)}(z|x_t)) + \mathbb{H}(p_\theta^{(t)}(x_t))$. *Since* $\mathbb{H}(p_\theta^{(t)}(x_t, z))$ *is constant with respect to* θ*, it follows that:*

$$\max_\theta \mathbb{H}(p_\theta^{(t)}(x_t)) = \min_\theta \mathbb{H}(p_\theta^{(t)}(z|x_t)) \leq \min_{\theta,\phi} -\iint p_\theta^{(t)}(x_t, z) \log q_\phi^{(t)}(z|x_t)\, \mathrm{d}z\, \mathrm{d}x_t, \tag{18}$$

where $q_\phi^{(t)}(z|x_t)$ *is a variational approximation to the true posterior* $p_\theta^{(t)}(z|x_t)$.

See Appendix E for a proof. This result mirrors the variational approach in the Information Maximisation (IM) algorithm [3], which maximises mutual information by minimising the conditional entropy using a variational decoder. Hence, this entropy bound can be viewed as a continuous latent-variable adaptation of the IM algorithm, where entropy maximisation serves to promote informative and diverse model outputs.

4 Experiments

To demonstrate the effectiveness of our approach, we consider a toy 2d mixture of 40 Gaussians (MoG-40) proposed in [30], where the means are uniformly distributed over $[-40, 40]^2$. We train VarDiU with two types of posteriors: **(a)** a Gaussian posterior with learnable mean and variance (VarDiU-Gaussian), and **(b)** a Neural Spline Flow (NSF; [9]) with a learnable-parameters Gaussian base distribution (VarDiU-NSF). We compare VarDiU to Diff-Instruct [28] using varying numbers of student score training steps $(1, 5, 10)$. Experimental details can be found in Appendix F.

Settings We consider three practical settings: **(1)** access to the true analytical score; **(2)** access to the training dataset; **(3)** access to a teacher score pre-trained by a diffusion model. For the given training

Table 1: **Comparison across different settings.** (↑) denotes the higher is better and (↓) denotes the lower is better. Best performance is in bold. Second best is underlined. *"True"* indicates the samples from the target distribution. Diff-inst (k) means Diff-instruct trained with k score steps. We can find VarDiU achieve best results with true and empirical score and remain competitive with learned score.

(a) Comparison with true score

Metrics	*True*	Diff-Inst (1)	Diff-Inst (5)	Diff-Inst (10)	VarDiU Gaussian	VarDiU NSF
Log-Density (↑)	*-6.65*	-10.17 ± 1.77	-8.09 ± 0.86	-7.86 ± 0.56	$\underline{-7.00 \pm 0.02}$	$\mathbf{-6.99 \pm 0.01}$
Log-MMD (↓)	/	-5.59 ± 0.36	-6.09 ± 0.40	-5.79 ± 0.36	$\underline{-6.86 \pm 0.29}$	$\mathbf{-7.33 \pm 0.17}$

(b) Comparison with training data

Metrics	*True*	Diff-Inst (1)	Diff-Inst (5)	Diff-Inst (10)	VarDiU Gaussian	VarDiU NSF
Log-Density (↑)	*-6.65*	-9.65 ± 0.62	-8.38 ± 0.42	-7.96 ± 0.62	$\underline{-7.09 \pm 0.03}$	$\mathbf{-7.01 \pm 0.02}$
Log-MMD (↓)	/	-5.51 ± 0.26	-5.60 ± 0.30	-5.63 ± 0.68	$\underline{-6.36 \pm 0.22}$	$\mathbf{-6.81 \pm 0.30}$

(c) Comparison with learned score

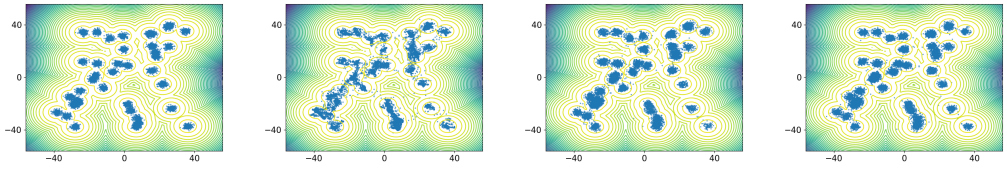
Metrics	*True*	EDM	Diff-Inst (1)	Diff-Inst (5)	Diff-Inst (10)	VarDiU Gaussian	VarDiU NSF
Log-Density (↑)	*-6.65*	-6.69	-10.88 ± 1.55	-9.21 ± 0.70	-8.47 ± 0.56	$\underline{-8.13 \pm 0.19}$	$\mathbf{-7.89 \pm 0.15}$
Log-MMD (↓)	/	-6.67	-5.24 ± 0.30	-5.57 ± 0.27	$\mathbf{-5.82 \pm 0.27}$	-5.64 ± 0.25	$\underline{-5.68 \pm 0.30}$

data case, we use the empirical score, defined by $\nabla_{x_t} \log \hat{p}_d(x_t) = \frac{1}{N} \sum_{n=1}^{N} \nabla_{x_t} \log q(x_t|x_0^{(n)})$ with the given dataset $\mathcal{D} = \{x_0^{(n)}\}_{n=1}^N$, where $q(x_t|x_0)$ matches the diffusion schedule of the VarDiU. This empirical score is a consistent estimator of the true data score [42, 43, 56], see Appendix F.3 for details. The learned score estimator is obtained by training EDM [17], which is a popular class of diffusion models, on the observed data. For each setting, we report results over 10 independent runs. The noise schedule and weighting strategy are detailed in Appendix F.2.

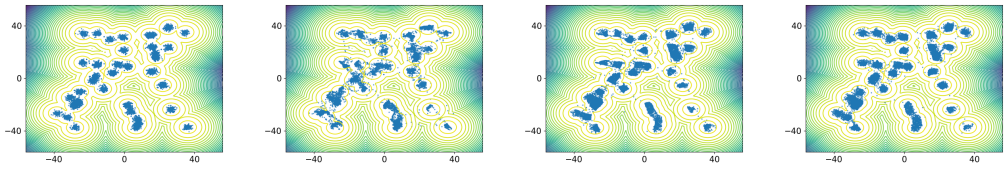
Metrics We report the *log-density* of generated samples under the true data distribution, which reflects sample quality but not sample diversity. We additionally compute the Maximum Mean Discrepancy (MMD; [12]) with five kernels which can measure diversity, see for details in Appendix F.1.

Sample Quality We compare samples generated by one-step models trained with VarDiU and Diff-Instruct across three practical settings: using the true score, the training dataset, and a learned score (see Figure 1 for visualizations). For Diff-Instruct, we train the model with 10 score-matching steps per generator gradient step to obtain the best sample quality we can, whereas the original Diff-Instruct paper [28] uses only a single step. Both VarDiU-Gaussian and VarDiU-NSF produce cleaner and sharper samples than Diff-Instruct when using the true score or the training dataset. With a learned score, however, all methods yield different sub-optimal results due to biases in the provided scores. We also compare log-density and log-MMD across the samples (see Tables 1a to 1c). We can find VarDiU consistently performs best under reliable evaluations (true score and training data), demonstrating high accuracy and stability with tight confidence intervals, while Diff-Instruct lags despite larger steps. Under learned scores, evaluations are noisier: VarDiU achieves higher log-density but slightly worse MMD, making comparisons unreliable due to bias in the given pre-trained scores. Overall, VarDiU is more accurate and stable across settings.

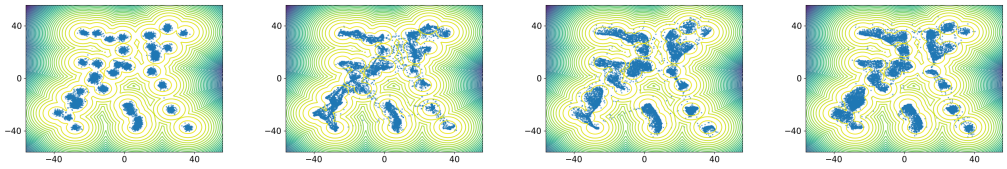
Training Stability and Efficiency We present plots of log-MMD against both generator gradient iterations and total training time in Figure 2; additional results are provided in Appendix Figures 4 to 6. Each curve shows the mean of 10 independent runs, with shaded regions indicating one standard deviation. For VarDiU, we apply a noise scheme with an annealing schedule. For Diff-Instruct, we use both the same annealing schedule (denoted as Diff-Instruct-a) and a non-annealing variant (see Appendix F.2 for details). In the log-MMD vs. generator gradient step plot, we find that under the same generator gradient budget (1M steps), VarDiU methods significantly outperform Diff-Instruct in both MMD values and stability (smaller variance). Moreover, incorporating a normalizing flow further improves performance compared to the Gaussian posterior, highlighting the importance of accurate generator gradient estimation. In the log-MMD vs. training time plot, we observe that under the same training hours, VarDiU-Gaussian clearly outperforms all other methods.



(a) True Samples (b) Diff-Instruct (c) VarDiU Gaussian (d) VarDiU NSF



(e) Empirical EDM (f) Diff-Instruct (g) VarDiU Gaussian (h) VarDiU NSF



(i) Pre-trained EDM (j) Diff-Instruct (k) VarDiU Gaussian (l) VarDiU NSF

Figure 1: The three rows correspond to the true score, training data, and learned score settings. Diff-Instruct models use 10 score-matching steps per generator step. Empirical EDM indicates the samples generated by EDM sampler with empirical scores. See Tables 1a to 1c for evaluation results.

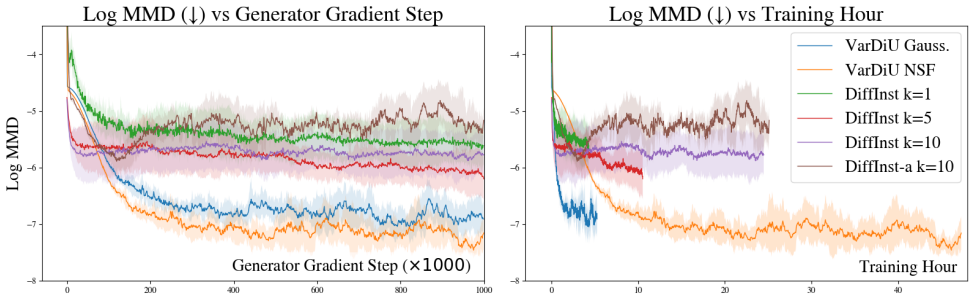


Figure 2: The left figure shows log-MMD over 1M generator steps, reflecting the efficiency of generator gradient estimation across different methods. The right figure shows total training time for 1M generator steps. In both, VarDiU clearly surpasses Diff-Instruct in quality, stability, and efficiency.

5 Conclusion and Future Work

In this paper, we introduced a novel method for diffusion distillation by proposing a variational diffusive upper bound on the DiKL divergence whose gradient can be estimated without bias. Our bound extends the Reverse KL upper bound from [56] to the diffusion setting and further eliminates the need to estimate the joint entropy. We also adapt this bound naturally to scenarios where only the teacher model's score function is accessible. Additionally, we establish a connection between our method and the information maximisation (IM [3]) framework, see Appendix G for a discussion on related works. Empirically, we demonstrate on a toy problem that our method achieves improved one-step generation quality while significantly reducing training time.

As future work, we are extending this approach to higher-dimensional data, such as image or video generation, which we believe to be a promising direction.

References

[1] F. Bao, C. Li, J. Sun, J. Zhu, and B. Zhang. Estimating the optimal covariance with imperfect mean in diffusion probabilistic models. *arXiv preprint arXiv:2206.07309*, 2022.

[2] F. Bao, C. Li, J. Zhu, and B. Zhang. Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. *arXiv preprint arXiv:2201.06503*, 2022.

[3] D. Barber and F. Agakov. The im algorithm: a variational approach to information maximization. *Advances in neural information processing systems*, 2004.

[4] D. Berthelot, A. Autef, J. Lin, D. A. Yap, S. Zhai, S. Hu, D. Zheng, W. Talbott, and E. Gu. Tract: Denoising diffusion models with transitive closure time-distillation. *arXiv preprint arXiv:2303.04248*, 2023.

[5] V. D. Bortoli, A. Galashov, J. S. Guntupalli, G. Zhou, K. Murphy, A. Gretton, and A. Doucet. Distributional diffusion models with scoring rules, 2025.

[6] Z. Botev and A. Ridder. Variance reduction. *Wiley statsRef: Statistics reference online*, 136:476, 2017.

[7] V. De Bortoli, J. Thornton, J. Heng, and A. Doucet. Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 2021.

[8] L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[9] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. Neural spline flows. *Advances in Neural Information Processing Systems*, 2019.

[10] B. Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 2014.

[12] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The journal of machine learning research*, (1):723–773, 2012.

[13] J. He, W. Chen, M. Zhang, D. Barber, and J. M. Hernández-Lobato. Training neural samplers with reverse diffusive kl divergence. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2025.

[14] J. Heek, E. Hoogeboom, and T. Salimans. Multistep consistency models. *arXiv preprint arXiv:2403.06807*, 2024.

[15] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 2020.

[16] F. Huszár. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.

[17] T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 2022.

[18] D. Kim, C.-H. Lai, W.-H. Liao, N. Murata, Y. Takida, T. Uesaka, Y. He, Y. Mitsufuji, and S. Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *International Conference on Learning Representations*, 2024.

[19] I. Kobyzev, S. J. Prince, and M. A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

[20] D. P. Kroese, T. Taimre, and Z. I. Botev. *Handbook of monte carlo methods*. John Wiley & Sons, 2013.

[21] L. Li and J. He. Bidirectional consistency models. *arXiv preprint arXiv:2403.18035*, 2024.

[22] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

[23] L. Liu, Y. Ren, Z. Lin, and Z. Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022.

[24] X. Liu, C. Gong, et al. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

[25] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 2022.

[26] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, pages 1–22, 2025.

[27] S. Luo, Y. Tan, L. Huang, J. Li, and H. Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

[28] W. Luo, T. Hu, S. Zhang, J. Sun, Z. Li, and Z. Zhang. Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. *Advances in Neural Information Processing Systems*, 2023.

[29] C. Meng, R. Rombach, R. Gao, D. Kingma, S. Ermon, J. Ho, and T. Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023.

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

[31] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*. PMLR, 2021.

[32] Z. Ou, M. Zhang, A. Zhang, T. Z. Xiao, Y. Li, and D. Barber. Improving probabilistic diffusion models with optimal covariance matching. *International Conference on Learning Representations*, 2025.

[33] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*. PMLR, 2015.

[34] H. E. Robbins. An empirical bayes approach to statistics. In *Breakthroughs in Statistics: Foundations and basic theory*, pages 388–394. Springer, 1992.

[35] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022.

[36] T. Salimans and J. Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

[37] T. Salimans, T. Mensink, J. Heek, and E. Hoogeboom. Multistep distillation of diffusion models via moment matching. *arXiv preprint arXiv:2406.04103*, 2024.

[38] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*. PMLR, 2015.

[39] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[40] Y. Song and P. Dhariwal. Improved techniques for training consistency models. In *International Conference on Learning Representations*, 2024.

[41] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency models. In *International Conference on Machine Learning*. PMLR, 2023.

[42] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 2019.

[43] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[44] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. 2021.

[45] V. Stimper, D. Liu, A. Campbell, V. Berenz, L. Ryll, B. Schölkopf, and J. M. Hernández-Lobato. normflows: A pytorch package for normalizing flows. *Journal of Open Source Software*, 8(86):5361, 2023.

[46] E. G. Tabak and C. V. Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

[47] E. G. Tabak and E. Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Commun. Math. Sci.*, 8(1):217–233, 2010.

[48] P. Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[49] Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, and J. Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 2024.

[50] Z. Xiao, K. Kreis, and A. Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.

[51] S. Xie, Z. Xiao, D. Kingma, T. Hou, Y. N. Wu, K. P. Murphy, T. Salimans, B. Poole, and R. Gao. Em distillation for one-step diffusion models. *Advances in Neural Information Processing Systems*, 2024.

[52] Y. Xu, W. Nie, and A. Vahdat. One-step diffusion models with f-divergence distribution matching. *arXiv preprint arXiv:2502.15681*, 2025.

[53] T. Yin, M. Gharbi, T. Park, R. Zhang, E. Shechtman, F. Durand, and B. Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in neural information processing systems*, 2024.

[54] L. Yu, T. Xie, Y. Zhu, T. Yang, X. Zhang, and C. Zhang. Hierarchical semi-implicit variational inference with application to diffusion model acceleration. *Advances in Neural Information Processing Systems*, 2024.

[55] S. Zhai, R. Zhang, P. Nakkiran, D. Berthelot, J. Gu, H. Zheng, T. Chen, M. A. Bautista, N. Jaitly, and J. Susskind. Normalizing flows are capable generative models. *arXiv preprint arXiv:2412.06329*, 2024.

[56] M. Zhang, T. Bird, R. Habib, T. Xu, and D. Barber. Variational f-divergence minimization. *arXiv preprint arXiv:1907.11891*, 2019.

[57] M. Zhang, W. Chen, J. He, Z. Ou, J. M. Hernández-Lobato, B. Schölkopf, and D. Barber. Towards training one-step diffusion models without distillation. *arXiv preprint arXiv:2502.08005*, 2025.

[58] M. Zhang, P. Hayes, T. Bird, R. Habib, and D. Barber. Spread divergence. In *International Conference on Machine Learning*. PMLR, 2020.

[59] L. Zhou, S. Ermon, and J. Song. Inductive moment matching. In *International Conference on Machine Learning*, 2025.

[60] M. Zhou, H. Zheng, Y. Gu, Z. Wang, and H. Huang. Adversarial score identity distillation: Rapidly surpassing the teacher in one step. In *International Conference on Learning Representations*, 2025.

[61] M. Zhou, H. Zheng, Z. Wang, M. Yin, and H. Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In *International Conference on Machine Learning*, 2024.

Appendix for "VarDiU: A Variational Diffusive Upper Bound for One-Step Diffusion Distillation"

Contents

A Proof of Variational Upper Bound

Proposition A.1 (Reverse KL Variational Upper Bound [56])**.** *For any choice of auxiliary variational distribution* $q_\phi^{(t)}(z \mid x_t)$*,*

$$\mathrm{KL}\Big(p_\theta^{(t)}(x_t) \,\|\, p_d^{(t)}(x_t)\Big) \le \mathrm{KL}\Big(p_\theta^{(t)}(x_t \mid z)p(z) \,\|\, p_d^{(t)}(x_t)q_\phi^{(t)}(z \mid x_t)\Big) =: \mathrm{U}^{(t)}(\theta, \phi). \tag{19}$$

Moreover, equality holds iff $q_\phi^{(t)}(z \mid x_t) = p_\theta^{(t)}(z \mid x_t)$ *for* $p_\theta^{(t)}(x_t)$*-almost every* x_t*.*

Proof. We start from the joint KL:

$$\mathrm{KL}\Big(p_\theta^{(t)}(x_t \mid z)p(z) \,\Big\|\, p_d^{(t)}(x_t)\, q_\phi^{(t)}(z \mid x_t)\Big) \tag{20}$$

$$= \iint p_\theta^{(t)}(x_t \mid z)p(z)\, \log \frac{p_\theta^{(t)}(x_t \mid z)p(z)}{p_d^{(t)}(x_t)\, q_\phi^{(t)}(z \mid x_t)}\, \mathrm{d}x_t\, \mathrm{d}z \tag{21}$$

$$= \iint p_\theta^{(t)}(x_t \mid z)p(z) \left[\log p_\theta^{(t)}(x_t \mid z) - \log p_d^{(t)}(x_t)\right] \mathrm{d}x_t\, \mathrm{d}z + \iint p_\theta^{(t)}(x_t \mid z)p(z) \left[\log p(z) - \log q_\phi^{(t)}(z \mid x_t)\right] \mathrm{d}x_t\, \mathrm{d}z \tag{22}$$

$$= \int p_\theta^{(t)}(x_t)\, \log \frac{p_\theta^{(t)}(x_t)}{p_d^{(t)}(x_t)}\, \mathrm{d}x_t + \iint p_\theta^{(t)}(x_t \mid z)p(z)\, \log \frac{p_\theta^{(t)}(z \mid x_t)}{q_\phi^{(t)}(z \mid x_t)}\, \mathrm{d}x_t\, \mathrm{d}z \tag{23}$$

$$= \mathrm{KL}\Big(p_\theta^{(t)}(x_t) \,\|\, p_d^{(t)}(x_t)\Big) + \mathbb{E}_{x_t \sim p_\theta^{(t)}(x_t)}\Big[\mathrm{KL}\Big(p_\theta^{(t)}(z \mid x_t) \,\|\, q_\phi^{(t)}(z \mid x_t)\Big)\Big]. \tag{24}$$

Since the conditional KL is non-negative, it follows that

$$\mathrm{KL}\Big(p_\theta^{(t)}(x_t)\,\|\,p_d^{(t)}(x_t)\Big) \le \mathrm{KL}\Big(p_\theta^{(t)}(x_t \mid z)p(z)\,\|\,p_d^{(t)}(x_t)q_\phi^{(t)}(z \mid x_t)\Big) =: \mathrm{U}^{(t)}(\theta,\phi). \tag{25}$$

Equality holds iff $q_\phi^{(t)}(z \mid x_t) = p_\theta^{(t)}(z \mid x_t)$ for $p_\theta^{(t)}(x_t)$ almost every x_t. □

B Proof of VarDiU Gradient Estimator

Proposition B.1. *The gradient can be estimated by*

$$\nabla_\theta \int p_\theta^{(t)}(x_t)\log p_d^{(t)}(x_t)\mathrm{d}x_t = \nabla_\theta \int p_\theta^{(t)}(x_t)\left(x_t^\top[\nabla_{x_t}\log p_d^{(t)}(x_t)]_{sg}\right)\mathrm{d}x_t, \tag{26}$$

where $[\,\cdot\,]_{\mathrm{sg}}$ *denotes the stop-gradient operator.*

Proof. Since x_t is generated via a differentiable transformation

$$x_t = \mathcal{F}_\theta(g_\theta(z),\boldsymbol{\epsilon},t) = g_\theta(z) + \boldsymbol{\epsilon}\sigma_t, \tag{27}$$

where $\boldsymbol{\epsilon}$ is drawn from a noise distribution $p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})$ that is independent of θ. Then we have

$$\begin{aligned}&\iint p_\theta^{(t)}(x_t|z)p(z)\log p_d^{(t)}(x_t)\,\mathrm{d}x_t\,\mathrm{d}z\\ &= \iint p(z)p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})\log p_d^{(t)}\Big(\mathcal{F}_\theta(g_\theta(z),\boldsymbol{\epsilon},t)\Big)\mathrm{d}\boldsymbol{\epsilon}\mathrm{d}z = \mathbb{E}_{z\sim p(z),\,\boldsymbol{\epsilon}\sim p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})}\Big[\log p_d^{(t)}\Big(\mathcal{F}_\theta(g_\theta(z),\boldsymbol{\epsilon},t)\Big)\Big]\end{aligned} \tag{28}$$

Since $\mathcal{F}_\theta(g_\theta(z),\boldsymbol{\epsilon},t)$ is differentiable in θ and under appropriate regularity conditions, we have

$$\nabla_\theta \iint p_\theta^{(t)}(x_t|z)p(z)\log p_d^{(t)}(x_t)\mathrm{d}z\mathrm{d}x_t = \mathbb{E}_{z\sim p(z),\,\boldsymbol{\epsilon}\sim p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})}\left[\nabla_\theta \log p_d^{(t)}\big(\mathcal{F}_\theta(g_\theta(z),\boldsymbol{\epsilon},t)\big)\right] \tag{29}$$

Using the chain rule, the gradient inside the expectation becomes

$$\nabla_\theta \log p_d^{(t)}\big(\mathcal{F}_\theta(g_\theta(z),\boldsymbol{\epsilon},t)\big) = \nabla_{x_t}\log p_d^{(t)}(x_t)\Big|_{x_t=\mathcal{F}_\theta(g_\theta(z),\boldsymbol{\epsilon},t)} \cdot \frac{\partial \mathcal{F}_\theta(g_\theta(z),\boldsymbol{\epsilon},t)}{\partial\theta} \tag{30}$$

Substituting back into our expression for the gradient, we obtain

$$\begin{aligned}&\nabla_\theta \iint p_\theta^{(t)}(x_t|z)p(z)\log p_d^{(t)}(x_t)\mathrm{d}x_t\mathrm{d}z\\ &= \mathbb{E}_{z\sim p(z),\,\boldsymbol{\epsilon}\sim p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})}\left[\nabla_{x_t}\log p_d^{(t)}(x_t)\Big|_{x_t=\mathcal{F}_\theta(g_\theta(z),\boldsymbol{\epsilon},t)} \cdot \frac{\partial \mathcal{F}_\theta(g_\theta(z),\boldsymbol{\epsilon},t)}{\partial\theta}\right] \qquad (31)\\ &= \mathbb{E}_{z\sim p(z),x_t\sim p_\theta^{(t)}(x_t|z)}\left[\nabla_{x_t}\log p_d^{(t)}(x_t)\frac{\partial x_t}{\partial\theta}\right] = \iint p_\theta^{(t)}(x_t|z)p(z)\nabla_{x_t}\log p_d^{(t)}(x_t)\frac{\partial x_t}{\partial\theta}\mathrm{d}x_t\mathrm{d}z, \qquad (32)\end{aligned}$$

as required. □

C A Practically Equivalent Variational Diffusive Upper Bound Loss

Tweedie's formula [10, 34]. For the Gaussian corruption kernel $p_t(x_t \mid x) = \mathcal{N}(x_t; x, \sigma_t^2\mathbf{I})$, Tweedie's identity gives

$$\nabla_{x_t}\log p_d^{(t)}(x_t) = \frac{\mu(x_t;t) - x_t}{\sigma_t^2}, \qquad \mu(x_t;t) := \mathbb{E}_{p_d(x|x_t)}[x \mid x_t]. \tag{33}$$

We have from Equation (12)

$$\begin{aligned}\mathcal{L}(\phi,\theta) &= -\int_0^1 \omega(t)p_\theta^{(t)}(x_t,z)\left[x_t^\top[\nabla_{x_t}\log p_d^{(t)}(x_t)]_{\mathrm{sg}} + \log q_\phi^{(t)}(z|x_t;t)\right]\mathrm{d}t + const\\ &= -\int_0^1 \omega(t)p_\theta^{(t)}(x_t,z)\left[g_\theta(z)^\top\left[\frac{\mu(x_t;t)-g_\theta(z)}{\sigma_t^2}\right]_{\mathrm{sg}} + \log q_\phi^{(t)}(z|x_t;t)\right]\mathrm{d}t + const.\end{aligned} \tag{34}$$

Proof. We are given the VarDiU objective:

$$\mathcal{L}(\phi,\theta) = -\int_0^1 \omega(t)\; p_\theta^{(t)}(x_t,z)\Big[x_t^\top\big[\nabla_{x_t}\log p_d^{(t)}(x_t)\big]_{\mathrm{sg}} + \log q_\phi^{(t)}(z\mid x_t;t)\Big]\mathrm{d}t + \mathrm{const}. \tag{35}$$

With the stop-gradient applied to the score term,

$$\begin{aligned} x_t^\top\big[\nabla_{x_t}\log p_d^{(t)}(x_t)\big]_{\mathrm{sg}} &= \frac{1}{\sigma_t^2}\,x_t^\top\big[\mu(x_t;t)-x_t\big]_{\mathrm{sg}} \\ &= \frac{1}{\sigma_t^2}\Big\{x^\top\big[\mu(x_t;t)-x\big]_{\mathrm{sg}} + (x_t-x)^\top\big[\mu(x_t;t)-x_t\big]_{\mathrm{sg}}\Big\}. \end{aligned} \tag{36}$$

Write $x_t = x + \sigma_t\varepsilon$ with $\varepsilon\sim\mathcal{N}(0,\mathbf{I})$, and note that $\big[\mu(x_t;t)-x_t\big]_{\mathrm{sg}}$ is treated as a constant w.r.t. (ϕ,θ). Taking the expectation over $x_t\mid x$ of (36) yields

$$\begin{aligned} &\mathbb{E}_{x_t|x}\Big[x_t^\top\big[\nabla_{x_t}\log p_d^{(t)}(x_t)\big]_{\mathrm{sg}}\Big] \\ &\qquad= \frac{1}{\sigma_t^2}\,\mathbb{E}_{x_t|x}\Big[x^\top\big[\mu(x_t;t)-x\big]_{\mathrm{sg}}\Big] + \frac{1}{\sigma_t^2}\,\mathbb{E}_{x_t|x}\Big[(x_t-x)^\top\big[\mu(x_t;t)-x_t\big]_{\mathrm{sg}}\Big]. \end{aligned} \tag{37}$$

The first term on the right is exactly $x^\top\big[\big(\mu(x_t;t)-x\big)/\sigma_t^2\big]_{\mathrm{sg}}$. The second term decomposes as

$$\mathbb{E}_{x_t|x}\Big[(x_t-x)^\top\big[\mu(x_t;t)\big]_{\mathrm{sg}}\Big] - \mathbb{E}_{x_t|x}\big[\|x_t-x\|^2\big].$$

Because $(x_t-x)=\sigma_t\varepsilon$ with ε independent of (ϕ,θ) and $\big[\mu(\cdot;t)\big]_{\mathrm{sg}}$ blocks all gradients through its argument, this entire bracket has *zero gradient* w.r.t. (ϕ,θ); in particular, $\mathbb{E}_{x_t|x}\big[\|x_t-x\|^2\big] = d\,\sigma_t^2$ (with d the data dimension) and $\mathbb{E}_{x_t|x}\big[(x_t-x)^\top[\mu(x_t;t)]_{\mathrm{sg}}\big]$ does not contribute to parameter gradients under reparametrisation. Hence it can be absorbed into "const".

Combining, we obtain (up to a parameter-independent constant)

$$\mathbb{E}_{x_t|x}\Big[x_t^\top\big[\nabla_{x_t}\log p_d^{(t)}(x_t)\big]_{\mathrm{sg}}\Big] = x^\top\Big[\frac{\mu(x_t;t)-x}{\sigma_t^2}\Big]_{\mathrm{sg}} + \mathrm{const}. \tag{38}$$

Substituting $x = g_\theta(z)$ and keeping the $\log q_\phi^{(t)}(z\mid x_t;t)$ term unchanged gives

$$\mathcal{L}(\phi,\theta) = -\int_0^1 \omega(t)\; p_\theta^{(t)}(x_t,z)\Big[g_\theta(z)^\top\Big[\frac{\mu(x_t;t)-g_\theta(z)}{\sigma_t^2}\Big]_{\mathrm{sg}} + \log q_\phi^{(t)}(z\mid x_t;t)\Big]\mathrm{d}t + \mathrm{const}, \tag{39}$$

as required. □

D Joint Entropy is Constant

Proposition D.1. *Let*

$$p_\theta^{(t)}(x,z) = p(z)\,p_\theta^{(t)}(x|z) \tag{40}$$

with a fixed prior $p(z)$ and conditional $\mathcal{N}\big(x;\,g_\theta(z),\sigma_t^2 I\big)$. Then the joint entropy $\mathbb{H}(p_\theta^{(t)}(x_t,z))$ is independent of θ.

Proof. First, write the joint entropy as the sum of marginal and conditional entropies:

$$\mathbb{H}(p_\theta^{(t)}(x_t,z)) = -\iint p_\theta^{(t)}(x_t,z)\;\log p_\theta^{(t)}(x_t,z)\,\mathrm{d}x_t\,\mathrm{d}z = \mathbb{H}(p(z)) + \mathbb{H}(p_\theta^{(t)}(x_t|z)), \tag{41}$$

where

$$\mathbb{H}(p(z)) = -\int p(z)\;\log p(z)\,\mathrm{d}z, \mathbb{H}(p_\theta^{(t)}(x_t|z)) = -\iint p(z)\,p_\theta^{(t)}(x_t|z)\;\log p_\theta^{(t)}(x_t|z)\,\mathrm{d}x_t\,\mathrm{d}z. \tag{42}$$

By assumption, the prior $p(z)$ does not depend on θ, so $\mathbb{H}(p(z))$ is a fixed number.

For each z, $x_t|z \sim \mathcal{N}(g_\theta(z), \sigma_t^2 I)$. The entropy of a Gaussian depends only on its covariance:

$$\mathbb{H}\big(\mathcal{N}(\mu, \Sigma)\big) = \tfrac{1}{2}\log\big[(2\pi e)^d \det \Sigma\big]. \tag{43}$$

where d is independent of θ, Here $\Sigma = \sigma_t^2 I$, so

$$\mathbb{H}\big(\mathcal{N}(g_\theta(z), \sigma_t^2 I)\big) = \tfrac{d}{2}\,\log\big[(2\pi e)\sigma^2\big], \tag{44}$$

which is independent of $g_\theta(z)$. Hence

$$\mathbb{H}(p_\theta^{(t)}(x_t|z)) = \int p(z)\,\mathbb{H}\big(\mathcal{N}(g_\theta(z), \sigma_t^2 I)\big)\,\mathrm{d}z = \tfrac{d}{2}\,\log\big[(2\pi e)\sigma_t^2\big], \tag{45}$$

also constant. Since both $\mathbb{H}(p(z))$ and $\mathbb{H}(p_\theta^{(t)}(x_t|z))$ are independent of θ, their sum $\mathbb{H}(p_\theta^{(t)}(x_t, z))$ is a constant. □

E Proof of Theorem 3.2

Theorem E.1 (Restatement of Theorem 3.2)**.** *By the chain rule of entropy, we have:* $\mathbb{H}(p_\theta^{(t)}(x_t, z)) = \mathbb{H}(p_\theta^{(t)}(z|x_t)) + \mathbb{H}(p_\theta^{(t)}(x_t))$. *Since* $\mathbb{H}(p_\theta^{(t)}(x_t, z))$ *is constant with respect to* θ*, it follows that:*

$$\max_\theta \mathbb{H}(p_\theta^{(t)}(x_t)) = \min_\theta \mathbb{H}(p_\theta^{(t)}(z|x_t)) \le \min_{\theta,\phi}\; -\iint p_\theta^{(t)}(x_t, z)\log q_\phi^{(t)}(z|x_t)\,\mathrm{d}z\,\mathrm{d}x_t, \tag{46}$$

where $q_\phi^{(t)}(z|x_t)$ *is a variational approximation to the true posterior* $p_\theta^{(t)}(z|x_t)$.

Proof. We have

$$\begin{aligned}
\mathbb{H}(p_\theta^{(t)}(z|x_t)) &= -\iint \log p_\theta(z|x_t) p_\theta(z|x_t) p_\theta^{(t)}(x_t)\mathrm{d}z\mathrm{d}x_t && (47)\\
&= -\iint \log\left(\frac{p_\theta(z|x_t)}{q_\phi^{(t)}(z|x_t)} q_\phi^{(t)}(z|x_t)\right) p_\theta^{(t)}(x_t|z)p(z)\mathrm{d}z\mathrm{d}x_t && (48)\\
&= -\iint \log \frac{p_\theta(z|x_t)}{q_\phi^{(t)}(z|x_t)} p_\theta^{(t)}(x_t|z)p(z)\mathrm{d}z\mathrm{d}x_t - \iint \log q_\phi^{(t)}(z|x_t) p_\theta^{(t)}(z|x_t) p_\theta^{(t)}(x_t)\mathrm{d}z\mathrm{d}x_t && (49)\\
&= -\underbrace{\mathrm{KL}(p_\theta(z|x_t)||q_\phi^{(t)}(z|x_t))}_{\ge 0} - \int \log q_\phi^{(t)}(z|x_t) p_\theta(z|x_t) p_\theta^{(t)}(x_t)\mathrm{d}z\mathrm{d}x_t. && (50)
\end{aligned}$$

Therefore,

$$\mathbb{H}(p_\theta^{(t)}(z|x_t)) \le -\iint \log q_\phi^{(t)}(z|x_t) p_\theta(z|x_t) p_\theta^{(t)}(x_t)\mathrm{d}z\mathrm{d}x_t$$

and the bound is tight when $\mathrm{KL}(p_\theta(z|x_t)||q_\phi^{(t)}(z|x_t)) = 0$. To minimise $\mathrm{KL}(p_\theta(z|x_t)||q_\phi^{(t)}(z|x_t))$ wrt ϕ, we have

$$\begin{aligned}
&\min_\phi \mathrm{KL}(p_\theta^{(t)}(z|x_t)||q_\phi^{(t)}(z|x_t)) = \min_\phi \iint \log q_\phi(z|x_t) p_\theta^{(t)}(x_t|z)p(z)\mathrm{d}z\mathrm{d}x_t.\\
\Rightarrow &\max_\theta \mathbb{H}(p_\theta^{(t)}(x_t)) = \min_\theta \mathbb{H}(p_\theta^{(t)}(z|x_t)) \le \min_\theta \min_\phi - \iint \log q_\phi^{(t)}(z|x_t) p(z) p_\theta^{(t)}(x_t|z)\mathrm{d}z\mathrm{d}x_t
\end{aligned} \tag{51}$$

where the last line since Proposition D.1

$$\underbrace{\mathbb{H}(p_\theta^{(t)}(x_t, z))}_{\text{const.}} = \mathbb{H}(p_\theta^{(t)}(x_t)) + \mathbb{H}(p_\theta^{(t)}(z|x_t)) \Rightarrow \mathbb{H}(p_\theta^{(t)}(x_t)) = -\mathbb{H}(p_\theta^{(t)}(z|x_t)) + \text{const.}$$

□

F Experimental Details

In this section, we present the details of the experimental setting.

F.1 Implementation Details

For the generator g_θ, we use a 5-layer MLP with latent dimension 2, hidden dimension 400 and SiLU activation. We use Adam to train the generator with learning rate 10^{-4}, batch size 1024, and gradient norm clip 10.0.

We employ the same 5-layer MLP with hidden dimension 400, and SiLU activation as well to predict the mean and log-variance of the Gaussian posterior. We use Adam to train the generator and posterior network with learning rate 10^{-4}, batch size 1024, and gradient norm clip 10.0.

For Diff-instruct [28], we use the same 5-layer MLP with hidden dimension 400, and SiLU activation to parametrise the student score network. We use Adam to train the student score network with learning rate $5 * 10^{-5}$, batch size 1024, and gradient norm clip 10.0.

Both methods are trained for $1{,}000{,}000$ epochs until convergence. For the learned score, we train an EDM [17] on $10{,}000$ observed samples, while the empirical score is estimated from $10{,}000$ observed samples. All experiments are conducted on `NVIDIA RTX 3090` GPUs.

Maximum Mean Discrepancy Implementation Details For evaluation, we employ the Maximum Mean Discrepancy (MMD) [12] with five Gaussian kernels of bandwidths $\{2^{-2}, 2^{-1}, 2^0, 2^1, 2^2\}$ for evaluation across multiple scales. Given two sets of samples $\{x_i\}_{i=1}^n \sim p_d$ and $\{y_j\}_{j=1}^m \sim p_\theta$, the squared MMD is defined as

$$\mathrm{MMD}^2(p_d, p_\theta; k) = \mathbb{E}_{x,x'\sim p_d}[k(x, x')] + \mathbb{E}_{y,y'\sim p_\theta}[k(y, y')] - 2\mathbb{E}_{x\sim p_d, y\sim p_\theta}[k(x, y)], \tag{52}$$

where $k(\cdot, \cdot)$ is the positive definite kernel. A smaller MMD indicates closer alignment between the generated and real distributions. For the results in Table 1, we perform 10 independent training runs. Evaluation metrics are recorded every $1{,}000$ epochs, and at each evaluation step, we compute the mean and standard deviation across runs. We use $10,000$ samples for MMD evaluation. To summarise performance, we report the average of these statistics over the last 50 evaluations.

VarDiU Posterior Parametrisation For VarDiU, we parametrise the Gaussian posterior as

$$q_\phi^{(t)}(z|x_t; t) = \mathcal{N}\Big(z;\, \mu_\phi(x_t; t),\, \sigma_t^2\, \mathrm{diag}\big(\sigma_\phi^2(x_t; t)\big)\Big). \tag{53}$$

The time-dependent scaling factor σ_t modulates the entropy of the base distribution to match the noise level at diffusion step t. This allows the posterior to capture fine-grained detail at small t, while maintaining flexibility to denoise highly corrupted samples at larger t.

For flow-based posterior, we implement Neural Splines Flows [9] using an open-source package `normflows`[1] [45], we choose the flow length to be 4. We use Adam to train the generator and flow-based posterior network with learning rate 10^{-4}, batch size 1024, and gradient norm clip 10.0 as well.

F.2 Noise Weighting and Scheduling

The one-step generation procedure is given by

$$x = \int \delta(x - g_\theta(z))\mathrm{d}x, \tag{54}$$

where the latent prior is sampled as $z \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{init}}^2 \boldsymbol{I})$. We set $\sigma_{\text{init}}^2 = 1$ in all toy data experiments.

For the forward diffusion distribution, we set

$$q(x_t|x_0) = \mathcal{N}(x_t; \alpha_t x_0, \sigma_t^2 I) \tag{55}$$

[1] https://github.com/VincentStimper/normalizing-flows

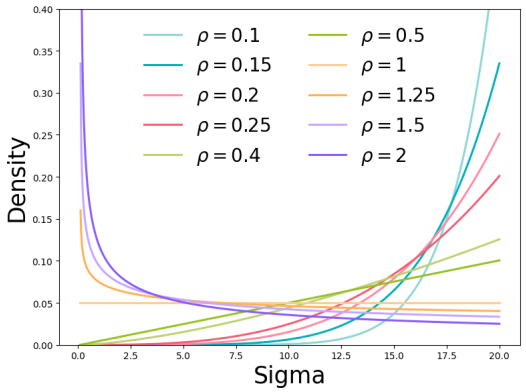


Figure 3: Density of sigma v.s different ρ. Here $\sigma_{\min} = 0.1, \sigma_{\max} = 20$.

Unlike Variance Preserving (VP;[42, 15]) schedule which must satisfy $\alpha_t^2 + \sigma_t^2 = 1$, we employ the schedule used in EDM [17] which is a popular class of diffusion models, where we set $\alpha_t = 1$ for all t and we sample the time variable $t \sim \text{Unif}[0, 1]$ and define the corresponding noise levels as

$$\sigma_t = \sigma_{\min} + t^{\rho} (\sigma_{\max} - \sigma_{\min}). \tag{56}$$

A visualisation of the resulting density of σ_t under different values of ρ is provided in Figure 3. For Diff-Instruct, we fix $\rho = 1.5$ across all three score estimation settings. We found an annealing ρ does not help with the training dynamics of Diff-Instruct, so we use a fixed schedule.

Annealing ρ. In VarDiU, we adopt an annealing strategy in which ρ gradually increases with training iterations. Specifically, ρ is initialised at $\rho_{\text{init}} = 0.1$ and incremented by 0.01 every 1000 epochs until it reaches ρ_{end}.

The annealed noise schedule plays a crucial role in stabilising training. The central idea is to initially place more emphasis on larger noise levels (σ), where the learning problem is smoother and less sensitive to estimation errors. As training progresses, the schedule gradually shifts towards smaller σ values, allowing the model to refine details and capture higher-frequency structures. This coarse-to-fine strategy not only mitigates instability at the early stage but also improves convergence and sample quality in later stages.

(1) Given The True Score. We set $\sigma_{\min} = 0.1$ and $\sigma_{\max} = 20$ for both Diff-Instruct and VarDiU. For VarDiU, we set $\rho_{\text{end}} = 5.0$.

(2) Given A Pre-trained DM. For Diff-Instruct, we set $\sigma_{\min} = 1.1$ and $\sigma_{\max} = 40$. For VarDiU, we set $\sigma_{\min} = 1.5$, $\sigma_{\max} = 40$, and $\rho_{\text{end}} = 2.0$.

(3) Given A Dataset. We employ $10{,}000$ samples for empirical score estimation, with details provided in Appendix F.3. For both Diff-Instruct and VarDiU, we set $\sigma_{\min} = 0.65$ and $\sigma_{\max} = 40$, and for VarDiU, we additionally set $\rho_{\text{end}} = 2.0$.

Noise Weighting. For both Diff-Instruct and VarDiU, we adopt the weighting function

$$\omega(t) = \frac{\sigma_t^2}{\sigma_{\max}^2}.$$

The noise schedule aligns with [57, 28]. We further divide the weighting by the $\sigma_{\max}$ for not changing the learning rate of training.

F.3 Empirical Score Estimation

Score of a Gaussian-smoothed empirical distribution. Let $\{x_0^{(n)}\}_{n=1}^N \subset \mathbb{R}^d$ be data points, and define the Gaussian kernel

$$\phi_\sigma(x) := (2\pi\sigma^2)^{-d/2} \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right).$$

The σ-smoothed empirical distribution is

$$\hat{p}_\theta(x) := \frac{1}{N}\sum_{n=1}^{N} \mathcal{N}\big(x|x_0^{(n)}, \sigma^2 I\big) = \frac{1}{N}\sum_{n=1}^{N} \phi_\sigma(x - x_0^{(n)}).$$

Lemma F.1 (Log-sum gradient identity)**.** *For positive functions* $\{a_n(x)\}_{n=1}^N$*,*

$$\nabla_x \log\Big(\sum_{n=1}^{N} a_n(x)\Big) = \sum_{n=1}^{N} w_n(x)\, \nabla_x \log a_n(x), \qquad w_n(x) := \frac{a_n(x)}{\sum_{m=1}^N a_m(x)}.$$

Proof. By the chain and quotient rules,

$$\nabla_x \log\Big(\sum_n a_n\Big) = \frac{\sum_n \nabla_x a_n}{\sum_m a_m} = \sum_n \frac{a_n}{\sum_m a_m}\,\frac{\nabla_x a_n}{a_n} = \sum_n w_n\, \nabla_x \log a_n. \qquad \square$$

Applying the lemma with $a_n(x) = \phi_\sigma(x - x_0^{(n)})$ gives

$$\nabla_x \log \hat{p}_\theta(x) = \sum_{n=1}^{N} w_n(x;\sigma)\, \nabla_x \log \phi_\sigma(x - x_0^{(n)}),$$

with responsibilities

$$w_n(x;\sigma) = \frac{\exp\big(-\|x - x_0^{(n)}\|^2/(2\sigma^2)\big)}{\sum_{m=1}^N \exp\big(-\|x - x_0^{(m)}\|^2/(2\sigma^2)\big)}.$$

The Gaussian score is

$$\nabla_x \log \phi_\sigma(x - x_0^{(n)}) = -\frac{1}{\sigma^2}\big(x - x_0^{(n)}\big) = \frac{x_0^{(n)} - x}{\sigma^2}.$$

Therefore,

$$\nabla_x \log \hat{p}_\theta(x) = \sum_{n=1}^{N} w_n(x;\sigma)\, \frac{x_0^{(n)} - x}{\sigma^2} = \frac{\sum_{n=1}^N w_n(x;\sigma)\, x_0^{(n)} - x}{\sigma^2}.$$

Final form. For any query point x_t, the score of the Gaussian–KDE is

$$\nabla_{x_t} \log \hat{p}_\theta(x_t) = \frac{\bar{x}(x_t;\sigma) - x_t}{\sigma^2}\ ; \qquad \bar{x}(x;\sigma) := \sum_{n=1}^{N} w_n(x;\sigma)\, x_0^{(n)}. \tag{57}$$

We employ Equation (57) to compute the empirical score using the given dataset practically. The accuracy of empirical score estimation tends to be small when a large number of samples are given, providing useful teacher guidance.

F.4 Symmetric Sampling

To reduce the variance of the stochastic estimator, we employ a symmetric sampling scheme [6, 20] for both VarDiU and Diff-Instruct.

We first draw a half-batch of latent codes,

$$z^{(i)} \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{init}}\mathbf{I}), \quad i = 1, \ldots, \tfrac{B}{2},$$

where B is the batch size. We then duplicate the samples to construct a full batch,

$$z = \big(z^{(1)}, \ldots, z^{(B/2)},\ z^{(1)}, \ldots, z^{(B/2)}\big).$$

Passing these latents through the generator yields samples

$$x_0 = g_\theta(z).$$

Next, we simulate forward diffusion process

$$\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad i = 1, \ldots, \tfrac{B}{2},$$

and extend them symmetrically as

$$\boldsymbol{\epsilon} = \big(\boldsymbol{\epsilon}^{(1)}, \ldots, \boldsymbol{\epsilon}^{(B/2)}, \ -\boldsymbol{\epsilon}^{(1)}, \ldots, -\boldsymbol{\epsilon}^{(B/2)}\big).$$

Thus, each noise vector is paired with its negative, ensuring that the perturbations are exactly centred. We also sample noise levels from the power distribution

$$\sigma^{(i)} \sim p_\rho(\sigma), \quad i = 1, \ldots, \tfrac{B}{2},$$

and replicate them symmetrically:

$$\boldsymbol{\sigma} = \big(\sigma^{(1)}, \ldots, \sigma^{(B/2)}, \ \sigma^{(1)}, \ldots, \sigma^{(B/2)}\big).$$

Noisy inputs. The perturbed inputs are then constructed as

$$x_t = x_0 + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon},$$

where $\odot$ denotes element-wise multiplication.

We found symmetric sampling can reduce the variance during training, further enhance the training stability of VarDiU, and can help with stabilising the training with Diff-Instruct.

G Related Works

Diffusion acceleration has received significant attention as a means of reducing the number of steps required in the reverse process. At the core of this problem lies the posterior distribution $q(x_0 \mid x_t)$, which connects a noisy intermediate x_t to the original data x_0. Accurate modelling of this posterior enables faithful reconstruction of x_0 in fewer steps, thereby lowering the sampling cost. Existing approaches can be broadly grouped into strategies such as designing alternative samplers [25, 26, 39, 7], relaxing the Gaussian posterior assumption [5, 22, 24], or projecting the dynamics onto lower-dimensional subspaces [35]. Whilst these techniques substantially reduce sampling time, they still typically require a non-trivial number of steps to achieve high-quality generation.

Another important direction focuses on compressing the reverse diffusion process into shorter chains. This includes progressive distillation [36, 29], consistency models and their improvements [41, 40, 18], as well as extensions to latent diffusion [27, 35]. More recently, Inductive Moment Matching has been proposed as a single-stage framework that trains few-step models from scratch by matching distributional moments [59].

In parallel, several works have investigated single-step diffusion models. Some fine-tune pre-trained diffusion models by minimising divergences such as the Diffusive KL or Fisher divergence [28, 51, 61, 52], while others incorporate adversarial losses to enhance perceptual fidelity [53, 60]. Our work builds on this line by eliminating the need for a separate student score network, thereby avoiding bias from inaccurate score estimation and enabling more stable training with improved generation quality.

Finally, recent work has shown that teacher guidance is not strictly necessary for effective diffusion distillation, highlighting that the multi-scale representations learned by diffusion models are the crucial factor behind the success of one-step generators [57].

H More Convergence and Efficiency Results

We present the log-MMD plots with respect to both generator gradient iterations and total training time, using the pre-trained DM and the given dataset in Figures 5 and 6, respectively.

When relying on the pre-trained DM, both methods exhibit high variance and unstable curves due to inaccurate score predictions from the pre-trained model. The noticeable jump of the VarDiU variants in the middle of training arises from the annealing schedule, where the density concentrates on a range of σ values for which the pre-trained DM suffers from large estimation errors. Nevertheless, the VarDiU variants maintain relatively low variance in the log-density curves, highlighting their robustness.

In contrast, when using the given dataset, we estimate the score empirically. The empirical score becomes increasingly accurate with larger sample sizes. As shown in Figure 6, the VarDiU variants achieve performance comparable to that obtained with the true analytical score. Furthermore, VarDiU demonstrates substantially improved training stability and efficiency when employing a Gaussian-parameterised posterior.

Overall, these results highlight the superiority of VarDiU, which consistently achieves more stable and efficient convergence across different settings compared to the baseline.

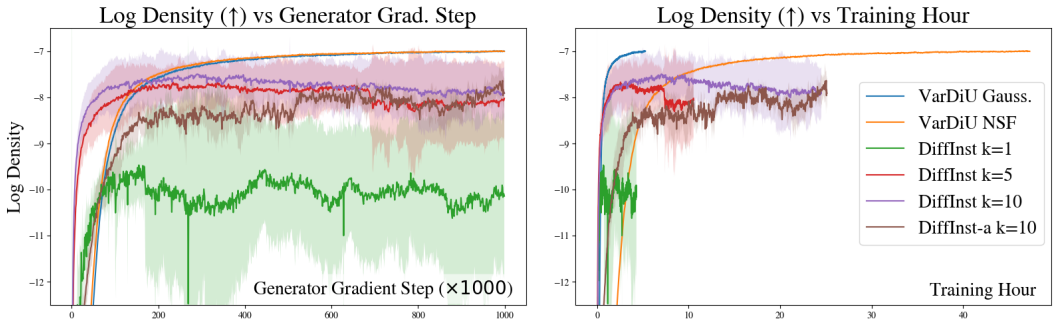
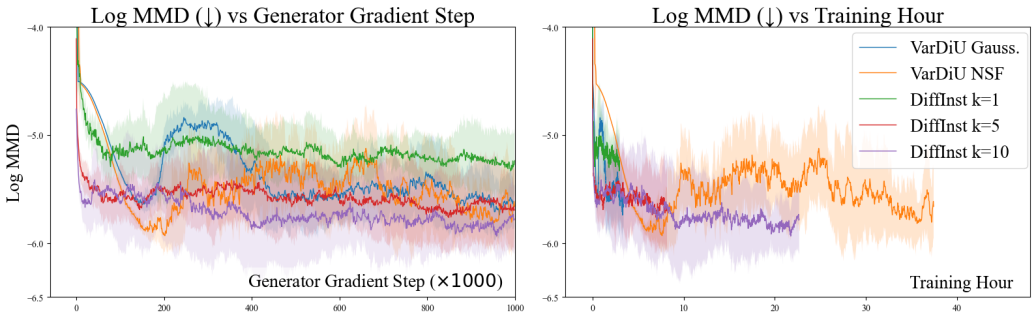
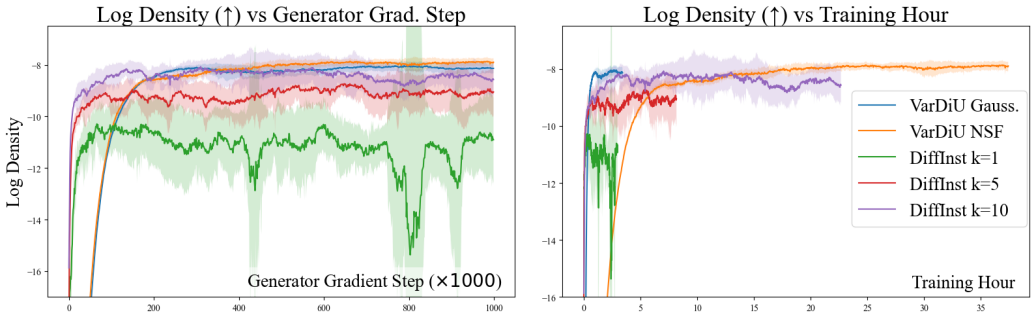


Figure 4: Log-density trajectory with true score seeting.

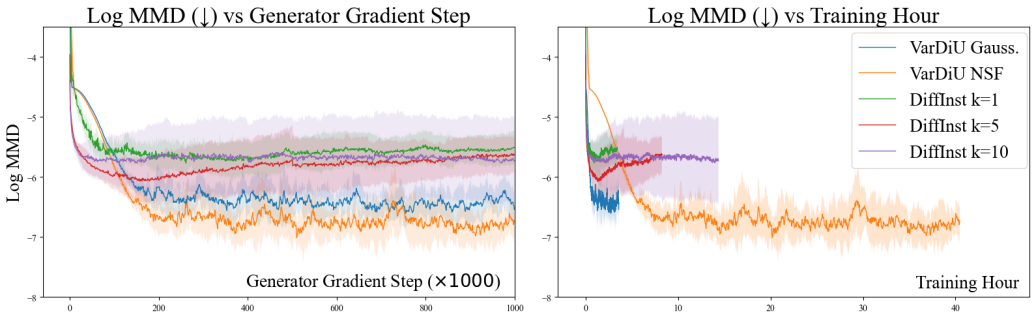


(a) Log-MMD trajectory with pre-trained DM.

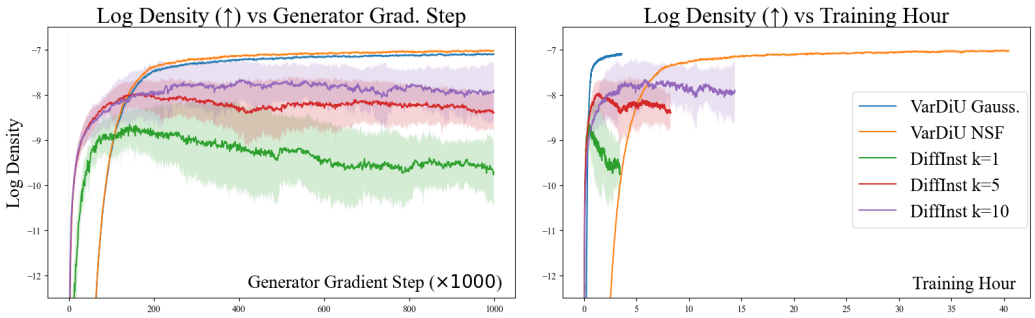


(b) Log-density trajectory with pre-trained DM..

Figure 5: Comparison of samples log-density and log-MMD trajectories with learned score.



(a) Log-MMD trajectory with a given dataset.



(b) Log-density trajectory with a given dataset.

Figure 6: Comparison of samples log-density and log-MMD trajectories under setting of given dataset.