On Inference Stability for Diffusion Models

Viet Nguyen[1*†], **Giang Vu**[2,3*], **Tung Nguyen Thanh**[2,3], **Khoat Than**[2‡], **Toan Tran**[1]

[1]VinAI Research, Vietnam
[2]Hanoi University of Science and Technology, Vietnam
[3] Viettel Group, Vietnam
{v.vietnv18, v.toantm3}@vinai.io, khoattq@soict.hust.edu.vn, {giangvl2, tungnt759}@viettel.com.vn

Abstract

Denoising Probabilistic Models (DPMs) represent an emerging domain of generative models that excel in generating diverse and high-quality images. However, most current training methods for DPMs often neglect the correlation between timesteps, limiting the model's performance in generating images effectively. Notably, we theoretically point out that this issue can be caused by the cumulative estimation gap between the predicted and the actual trajectory. To minimize that gap, we propose a novel *sequence-aware* loss that aims to reduce the estimation gap to enhance the sampling quality. Furthermore, we theoretically show that our proposed loss function is a tighter upper bound of the estimation loss in comparison with the conventional loss in DPMs. Experimental results on several benchmark datasets including CIFAR10, CelebA, and CelebA-HQ consistently show a remarkable improvement of our proposed method regarding the image generalization quality measured by FID and Inception Score compared to several DPM baselines. Our code and pre-trained checkpoints are available at https://github.com/VinAIResearch/SA-DPM.

Introduction

Diffusion Probabilistic Models (DPMs) (Sohl-Dickstein et al. 2015), inspired by statistical physics, have been shown to be more effective generative models than prior ones. Typically, a DPM consists of two processes: a forward process that gradually adds noise to the original data distribution and a reverse process that learns to iteratively reconstruct a data instance from the noises. As a progress of that idea, (Ho, Jain, and Abbeel 2020) proposes Denoising Diffusion Probabilistic Models (DDPMs) which exploit the knowledge about the transition distribution to derive the loss function and guide the training process. Parallel to that work, (Song and Ermon 2019) uses the score-based model to train a similar model. More recently, (Song et al. 2021) interprets those two works under the lens of stochastic differential equations. This class of models outperforms prior ones in terms of generated images' quality and distribution coverage. While other likelihood-based generative models require unique assumptions on data (Germain et al. 2015; Van den Oord et al. 2016) or constraints in model architecture (Dinh, Sohl-Dickstein, and Bengio 2017; Papamakarios, Pavlakou, and Murray 2017; Kingma and Dhariwal 2018; Ho et al. 2019) to perform well, DPMs do not hold any of that requirements. Moreover, compared with Generative Adversarial Networks, Diffusion Models do not require adversarial training thus making the learning process easy and stable.

Although DPMs have been shown to achieve state-of-the-art results in various data generation tasks since their debut, these models often suffer from slow sampling speed, which may require thousands of model feeds to achieve high sample quality. To address this issue, many researchers have focused on accelerating the generating process. For example, (Song, Meng, and Ermon 2021; Kong and Ping 2021) propose non-Markovian diffusion processes, which allow taking multiple steps at once to accelerate the sampling time. Several works explore finding short sampling trajectories by applying search algorithms, e.g., grid search (Chen et al. 2021), dynamic programming (Watson et al. 2021), and differentiable search (Watson et al. 2022). (Salimans and Ho 2022; Song et al. 2023) propose to boost the sampling process via knowledge distillation with the core idea of distilling a multi-step process into a single step.

(Song et al. 2021) establishes a connection between the denoising process and solving ordinary differential equations (ODE). Such a connection enables the use of numerical methods of differential equations to accelerate the denoising process. While (Song et al. 2021) proposes the use of higher-order solvers such as Runge-Kutta methods, (Liu et al. 2022) proposes pseudo-numerical methods to generate samples along a specific manifold. Another approach proposed by (Karras et al. 2022) is to use Heun's second-order method to solve the probability flow ODE.

Some recent attempts aim to refine inefficient sampling trajectories due to the approximation and optimization errors in training. (Bao et al. 2022b,a) propose to estimate the optimal variance to correct the potential bias caused by the imperfect mean estimation. Meanwhile, (Zhang, Niwa, and Kleijn 2023) introduces an extrapolation operation on two consecutive sampling steps to make the sampling trajectory closer to the direction of the real-data point.

One main drawback of those works is that they mostly focus on sampling efficiency by, for instance, making mod-

*These authors contributed equally, the order is random.
†This work was partly done while at HUST
‡Corresponding author

ifications in only the sampling process, or fine-tuning pre-trained DPMs, without training DPMs from scratch. In particular, we find out that most existing DPMs are often trained in a timestep-independence paradigm, which often ignores the sequential nature of DPMs in both forward and backward processes. We view the sampling trajectory at a global scale and derive the estimation gap of a noise predictor. That gap indicates how far the predicted trajectory is from the actual one. From that observation, we propose a new training objective, termed the *Sequence-Aware* (**SA**) loss, that constrains directly the gap. Our contributions are summarized below:

- We point out the estimation gap between the predicted and actual sampling trajectory and analyze its effect on the data generation quality of DPMs.
- We propose a novel *sequence-aware loss* and an induced training algorithm to minimize the estimation gap.
- We theoretically show that our loss function is a tighter upper bound of the estimation gap in comparison with the conventional loss function.
- We employ that loss in multiple DPM baselines. Empirical results illustrate significant improvements in FID and Inception Score compared to several current DPM baselines.

Background

Diffusion Probabilistic Models (Sohl-Dickstein et al. 2015) are comprised of two fundamental components, including the *forward process* and the *reverse process*. The former gradually diffuses each input $\boldsymbol{x}_0$, following a data distribution $q(\boldsymbol{x}_0)$, into a standard Gaussian noise through T timesteps, i.e., $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathbf{I}$ is the identity matrix, $\mathcal{N}(\cdot, \cdot)$ represents the normal distribution. The reverse process starts from $\boldsymbol{x}_T$ and then interactively denoises to get an original image. We recap the background of DPMs following the idea of DDPM (Ho, Jain, and Abbeel 2020).

Forward Process

Given an original data distribution $q(\boldsymbol{x}_0)$, the forward process can be presented as follows:

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}),$$

where $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \coloneqq \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{I})$ and an increasing noise scheduling sequence $\beta_t \in (0, 1]$, which describes the amount of noise added at each timestep t. Denoting $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, the distribution of diffused image $\boldsymbol{x}_t$ at timestep t has a closed form as:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I}).$$

By applying the reparameterization trick (Kingma and Welling 2013; Rezende, Mohamed, and Wierstra 2014), we can sample the data at each time step t by:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, \tag{1}$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The noise scheduler $\beta_{1:T}$ is designed in such a way that $\bar{\alpha}_{1:T}$ is a decreasing array and $\bar{\alpha}_T \approx 0$. That means at the end of the forward process, $\boldsymbol{x}_T$ is likely sampled from the standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

Algorithm 1: Conventional training

Require: Empirical data distribution q, number T of timesteps, the noise predictor $\boldsymbol{f}_\theta$, learning rate η.
repeat
  $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$
  $t \sim \text{Uniform}(\{1, \dots, T\})$
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$
  $\mathcal{L}_{simple} = \|\boldsymbol{f}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon}_t\|^2$
  $\theta \leftarrow \theta - \eta \bigtriangledown_\theta \mathcal{L}_{simple}$
until converged

Algorithm 2: Sampling

$\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
$\bar{\boldsymbol{x}}_T = \boldsymbol{x}_T$
for $t = T, \dots, 1$ **do**
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\boldsymbol{z} = 0$
  $\bar{\boldsymbol{x}}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\bar{\boldsymbol{x}}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{f}_\theta(\bar{\boldsymbol{x}}_t, t)) + \sigma_t\boldsymbol{z}$
end for
return $\bar{\boldsymbol{x}}_0$

Reverse Process

At each step of the forward diffusion process, only a small amount of Gaussian noise is added to the data. Therefore, the reverse conditional distribution $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ can be approximated by a Gaussian conditional distribution

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t, \boldsymbol{x}_0), \tilde{\beta}_t\mathbf{I}),$$

where $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$ and

$$\tilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t, \boldsymbol{x}_0) = \gamma_{1,t}\boldsymbol{x}_0 + \gamma_{2,t}\boldsymbol{x}_t, \tag{2}$$
$$\gamma_{1,t} = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}, \quad \gamma_{2,t} = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}.$$

Therefore, the trained denoising process $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ to approximate $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ can be parameterized by

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \sigma_t^2\mathbf{I}),$$

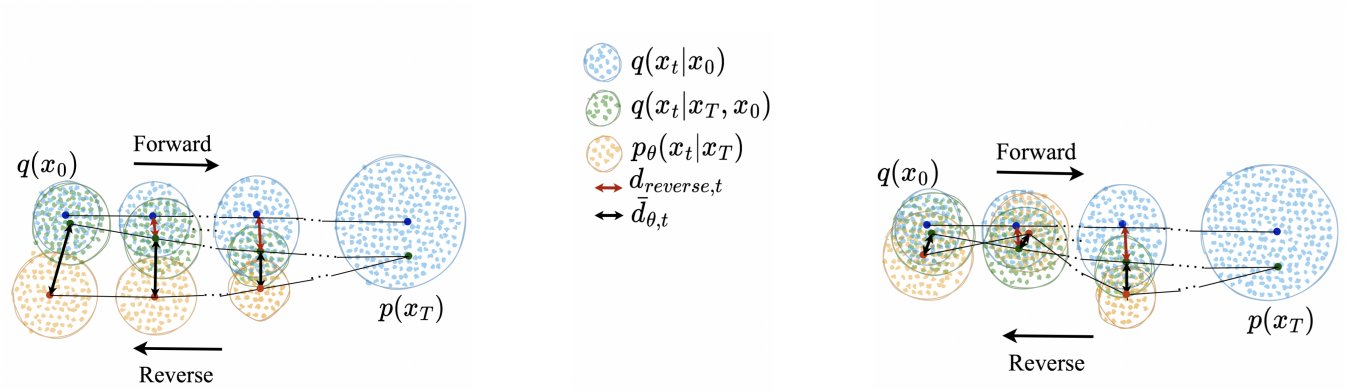
where $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ and $\sigma_t^2\mathbf{I}$ are the mean and covariance matrix of the parametric denoising model, respectively.

The training objective is then to maximize a variational lower bound on the log-likelihood of the original $\boldsymbol{x}_0$, which can be simplified (by excluding an additional term that is irrelevant to the training) as minimizing the loss:

$$\mathcal{L}(\theta) = -\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1) + \textstyle\sum_t D_{KL}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)||p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)).$$

The mean $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ predicted by the denoising model at each step can be reparameterized as a neural network that predicts the true $\boldsymbol{x}_0$. Alternately, following (Ho, Jain, and Abbeel 2020), one can use a noise prediction model $\boldsymbol{f}_\theta$ that predicts the noise $\boldsymbol{\epsilon}_t$ added to $\boldsymbol{x}_0$ to construct $\boldsymbol{x}_t$. This allows training by simply minimizing the mean squared error between the predicted noise $\boldsymbol{f}_\theta(\boldsymbol{x}_t, t)$ and the true added Gaussian noise $\boldsymbol{\epsilon}_t$ (detailed in Algorithm 1):

$$\mathcal{L}_{simple} = \mathbf{E}_{t, \boldsymbol{x}_0, \boldsymbol{\epsilon}_t}\big[\|\boldsymbol{f}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon}_t\|^2\big]. \tag{3}$$



(a) Large estimation gap $\bar{d}_\theta$

(b) Small estimation gap $\bar{d}_\theta$

Figure 1: 1-D example of sampling trajectory. Under the assumption that the error at each timestep is similar: (a) the cumulative error by steps is large while (b) the cumulative error by steps is small. This behavior is due to the correlation between neighbor timesteps.

After training, new samples can be generated by first sampling Gaussian noise $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then passing this noise through the trained model's iterative denoising procedure over T timesteps, ultimately outputting a new sample $\boldsymbol{x}_0$, detailed in Algorithm 2.

Methodology

In the sampling phase, a small amount of error may be introduced in each denoising iteration due to the imperfect learning process. Note that the inference process often requires many iterations to produce high-quality images, leading to the accumulation of these errors. In this section, we first point out the estimation gap between the predicted and ground-truth noises in the sampling process of DPMs and show its importance in the training phase to mitigate this accumulation and improve the quality of generated images. Based on that gap, we introduce a novel loss function that is proven to be tighter than $\mathcal{L}_{simple}$ commonly used in DPMs.

Estimation Gap

The data generation process in Diffusion Models is performed by iteratively sampling a datapoint from the predicted distribution of $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$. To interpret the working principle of the global trajectory, we take a further derivation on $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_T, \boldsymbol{x}_0)$, detailed in Appendix A, to obtain

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_T, \boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_t^{'}, \beta_t^{'}\mathbf{I}),$$
$$\text{where } \boldsymbol{\mu}_t^{'} = \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0 + \frac{\sqrt{\bar{\alpha}_T}(1-\bar{\alpha}_{t-1})}{\sqrt{\bar{\alpha}_{t-1}(1-\bar{\alpha}_T)}}\boldsymbol{\epsilon}_T.$$

Here, we can ignore the variance term since it is fixed in basic settings. We define $d_{reverse,t} = \frac{\sqrt{\bar{\alpha}_T}(1-\bar{\alpha}_{t-1})}{\sqrt{\bar{\alpha}_{t-1}(1-\bar{\alpha}_T)}}\boldsymbol{\epsilon}_T$ as the reverse gap term. As $\frac{\sqrt{\bar{\alpha}_T}(1-\bar{\alpha}_{t-1})}{\sqrt{\bar{\alpha}_{t-1}(1-\bar{\alpha}_T)}}$ decreases to 0 when t comes to the first step, the mean $\boldsymbol{\mu}_t^{'}$ converges to $\boldsymbol{x}_0$ naturally. In many real-life applications, at each timestep t, the sampling phase of DPMs aims to provide an approximation $\boldsymbol{x}_{\theta,0}^{(t)}$ of the true value $\boldsymbol{x}_0$ and the corresponding vector error $(\boldsymbol{x}_{\theta,0}^{(t)} - \boldsymbol{x}_0)$ is then expected to be sufficiently close to $\mathbf{0}$.

Technically, according to (2), the mean of the posterior distribution $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ at each timestep t is defined as: $\tilde{\boldsymbol{\mu}}_t = \gamma_{1,t}\boldsymbol{x}_0 + \gamma_{2,t}\boldsymbol{x}_t$. Note that $\boldsymbol{x}_t$ does not depend on the prediction $\boldsymbol{x}_{\theta,0}^{(t)}$. Given the true noise $\boldsymbol{\epsilon}_{1:T}$ added to $\boldsymbol{x}_0$, according to (1), the gap incurred by the noise predictor $\boldsymbol{f}_\theta(\boldsymbol{x}_t, t)$ at step t is defined as:

$$d_{\theta,t} = \gamma_{1,t}(\boldsymbol{x}_{\theta,0}^{(t)} - \boldsymbol{x}_0) = \gamma_{1,t}\frac{\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}(\boldsymbol{f}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon}_t). \tag{4}$$

Now we can formally point out the gap between the true noises and predictions by a model.

Theorem 1 (Estimation gap) *Let $\boldsymbol{f}_\theta(\boldsymbol{x}_s, s)$ be a noise predictor with parameter θ. Its total gap from step 2 to T, for each $\boldsymbol{x}_0$, is*

$$d_\theta(\boldsymbol{x}_0) = \sum_{i=2}^{T}\tau_i(\boldsymbol{f}_\theta(\boldsymbol{x}_i, i) - \boldsymbol{\epsilon}_i), \tag{5}$$

where $\tau_i = \frac{\sqrt{\bar{\alpha}_{i-1}}(1-\bar{\alpha}_1)}{\sqrt{\bar{\alpha}_1}(1-\bar{\alpha}_{i-1})}\gamma_{1,i}\frac{\sqrt{1-\bar{\alpha}_i}}{\sqrt{\bar{\alpha}_i}}$. Furthermore, the total loss of $\boldsymbol{f}_\theta$ is $\mathcal{L}_\theta = \mathbf{E}_{\boldsymbol{x}_0,\boldsymbol{\epsilon}}\|d_\theta(\boldsymbol{x}_0)\|^2$.

Proof sketch. Denote $\bar{d}_{\theta,T} = d_{\theta,T}$ and define $\bar{d}_{\theta,t} = d_{\theta,t} + \gamma_{2,t}\bar{d}_{\theta,t+1}$ to be the gap at an arbitrary timestep $t < T$. By induction (Appendix B), we have

$$\bar{d}_{\theta,t} = d_{\theta,t} + \sum_{i=t+1}^{T}\left[\prod_{s=t}^{i-1}\gamma_{2,s}\right]d_{\theta,i}$$
$$= d_{\theta,t} + \sum_{i=t+1}^{T}\left[\frac{\sqrt{\bar{\alpha}_{i-1}}(1-\bar{\alpha}_{t-1})}{\sqrt{\bar{\alpha}_{t-1}}(1-\bar{\alpha}_{i-1})}\right]d_{\theta,i}.$$

At the end of the trajectory, the estimation gap is

$$d_\theta = \bar{d}_{\theta,2} = \sum_{i=3}^{T} \left[\frac{\sqrt{\bar{\alpha}_{i-1}}(1-\bar{\alpha}_1)}{\sqrt{\alpha_1}(1-\bar{\alpha}_{i-1})} \right] d_{\theta,i} + d_{\theta,2}.$$

The proof is completed by using (4). ⊡

The term $d_\theta(\boldsymbol{x}_0)$ can be considered as the estimation gap of the model for each example $\boldsymbol{x}_0$, while $\mathcal{L}_\theta$ represents the overall *estimation error* which is critical for the training process. In typical DPMs, the training process is often performed by minimizing the conventional square loss $\mathcal{L}_{simple,t} = \|\boldsymbol{f}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon}_t\|^2$ at each step t, which may not necessarily minimize $\mathcal{L}_\theta$. It means that minimizing $\mathcal{L}_{simple}$ can produce multiple small gaps $d_{\theta,t}$. In the worst case, those small gaps can lead to a non-trivial total gap d_θ as visualized by a 1-D example in Figure 1a. Therefore, a better way to train a DPM is to directly minimize the total gap d_θ, instead of trying to minimize each independent term $\mathcal{L}_{simple,t}$. That scenario can be intuitively illustrated in Figure 1b.

Minimizing directly the whole d_θ is challenging due to the requirement of a large number of timesteps, which often leads to a significant memory and computation capability in the training phase. From that observation, we propose a new training loss that aims to minimize the gap term in a slice of trajectory. We name it *sequence-aware loss* based on the idea of considering the error amount of surrounding timesteps. In the next section, we introduce the new training loss and the training algorithm. We also theoretically show that any variants (based on the number of consecutive steps) of that loss function are a tighter upper bound of the estimation error compared to the conventional loss. Finally, we employ that loss function in multiple DPM frameworks and demonstrate its effectiveness on image generation quality.

Sequence-aware Training

Minimizing the mean squared error $||\boldsymbol{f}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon}_t||^2$ may lead to small gap value at each timestep. However, one critical issue of this approach is that it ignores the relationship between timesteps, which may cause a large total gap d_θ at the end of the trajectory. Instead of optimizing each individual term, minimizing the d_θ should guarantee a good approximation $p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_T)$ of the distribution $q(\boldsymbol{x}_0|\boldsymbol{x}_T)$. Nevertheless, that approach often requires a large amount of computation and memory. To address that issue, we propose to minimize the local gap that connects K consecutive steps (for $K > 1$):

$$d^K_{\theta,t} = \sum_{s=t}^{t+K-1} \tau_s(\boldsymbol{f}_\theta(\boldsymbol{x}_s, s) - \boldsymbol{\epsilon}_s).$$

The *sequence-aware (SA) loss* function for training is:

$$\mathcal{L}_{sa} = \mathbf{E}_{t,\boldsymbol{x}_0,\boldsymbol{\epsilon}_{t:t+K-1}} \left\| \frac{1}{K} \sum_{s=t}^{t+K-1} \tau_s(\boldsymbol{f}_\theta(\boldsymbol{x}_s, s) - \boldsymbol{\epsilon}_s) \right\|^2,$$

where $t \in \{1-K, ..., T\}$ and $\tau_s = 0$ for any $s \notin \{2, ..., T\}$. This training objective enforces the stability in the chain of K consecutive sampling steps. However, we found that optimizing that function independently makes the training error at each timestep quite large, since this SA loss does not strongly constrain the error at individual steps. Therefore, we suggest optimizing $\mathcal{L}_{sa}$ jointly with $\mathcal{L}_{simple}$ to exploit their advantages, resulting in the following total loss function for training DPMs:

$$\mathcal{L} = \mathcal{L}_{simple} + \lambda \mathcal{L}_{sa}, \tag{6}$$

where $\lambda \geq 0$ is a hyper-parameter that indicates how much we constrain the sampling trajectory. Optimizing the new loss term involves the direction of error at each step. Algorithm 3 represents the training procedure. In practice, we can ignore constants τ_s in $\mathcal{L}_{sa}$ since they are often comparable and empirically do not significantly change sample quality.

Algorithm 3: Sequence-aware training

Require: Data distribution q, number of timesteps T, the noise predictor $\boldsymbol{f}_\theta$, number of consecutive steps K, hyper-parameter λ, learning rate η.

repeat
- $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$
- $t \sim \text{Uniform}(\{1, \dots, T\})$
- **for** $k \in \{0, \dots, K-1\}$ **do**
  - $\boldsymbol{\epsilon}_{t+k} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - $\boldsymbol{x}_{t+k} = \sqrt{\bar{\alpha}_{t+k}}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_{t+k}}\boldsymbol{\epsilon}_{t+k}$
- **end for**
- $\mathcal{L}_{simple} = \|\boldsymbol{f}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon}_t\|^2$
- $\mathcal{L}_{sa} = \frac{1}{K^2}\| \sum_{s=t}^{t+K-1} \tau_s(\boldsymbol{f}_\theta(\boldsymbol{x}_s, s) - \boldsymbol{\epsilon}_s)\|^2$
- $\mathcal{L} = \mathcal{L}_{simple} + \lambda \mathcal{L}_{sa}$
- $\theta \leftarrow \theta - \eta \bigtriangledown_\theta \mathcal{L}$

until converged

Bounding the Estimation Gap

We have presented the new loss which incorporates more information of the sequential nature of DPMs. We next theoretically show that this loss is tighter than the vanilla loss.

Theorem 2 *Let $\boldsymbol{f}_\theta(\boldsymbol{x}_s, s)$ be any noise predictor with parameter θ. Consider the weighted conventional loss function $\mathcal{L}^\tau_{simple} \coloneqq \mathbf{E}_{t,\boldsymbol{x}_0,\boldsymbol{\epsilon}_t}\left[\tau_t^2 \|\boldsymbol{f}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon}_t\|^2\right]$, where τ_t is defined in Theorem 1 and $t \in \{2, ..., T\}$. Then*

$$\frac{T-1}{T+K}\mathcal{L}^\tau_{simple} \geq \mathcal{L}_{sa} \geq \frac{1}{(T+K)^2}\mathcal{L}_\theta. \tag{7}$$

Proof. By definition, $\tau_s = 0$ for any $s \notin \{2, ..., T\}$. We observe that:

$$\begin{aligned}
&(T-1)\mathcal{L}^\tau_{simple} \\
&= (T-1)\mathbf{E}_{t\in\{2,\dots,T\},\boldsymbol{x}_0,\boldsymbol{\epsilon}_t}\left[\tau_t^2 \|\boldsymbol{f}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon}_t\|^2\right] \\
&= \sum_{t=2}^{T} \mathbf{E}_{\boldsymbol{x}_0,\boldsymbol{\epsilon}_t}\left[\tau_t^2 \|\boldsymbol{f}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon}_t\|^2\right] \\
&= \sum_{t=1-K}^{T} \mathbf{E}_{\boldsymbol{x}_0,\boldsymbol{\epsilon}_{t:t+K-1}}\left[\frac{1}{K}\sum_{s=t}^{t+K-1} \tau_s^2 \|\boldsymbol{f}_\theta(\boldsymbol{x}_s, s) - \boldsymbol{\epsilon}_s\|^2\right]
\end{aligned}$$

Dataset	T	Method			
		DDPM		DDIM	
		B	SA	B	SA
CelebA-HQ 256 × 256	10	54.19	**53.23**	39.29	**37.66**
	50	29.04	**26.73**	23.04	**20.20**
	100	22.85	**20.66**	22.19	**18.98**
	200	18.71	**16.63**	22.52	**19.27**
	1000	16.03	**15.32**	24.10	**20.11**

Table 1: FID score (↓). The results are reported under different number T of timesteps. Here B and SA denote the baseline and our proposed loss.

Jensen's inequality suggests that

$$\frac{1}{K}\sum_s \tau_s^2 \left\|\boldsymbol{f}_\theta(\boldsymbol{x}_s, s) - \boldsymbol{\epsilon}_s\right\|^2 \geq \left\|\frac{1}{K}\sum_s \tau_s(\boldsymbol{f}_\theta(\boldsymbol{x}_s, s) - \boldsymbol{\epsilon}_s)\right\|^2.$$

Therefore, we have

$$\begin{aligned}&(T-1)\mathcal{L}^\tau_{simple}\\ &\geq \sum_{t=1-K}^{T} \mathbf{E}_{\boldsymbol{x}_0,\boldsymbol{\epsilon}_{t:t+K-1}} \left\|\frac{1}{K}\sum_{s=t}^{t+K-1} \tau_s(\boldsymbol{f}_\theta(\boldsymbol{x}_s, s) - \boldsymbol{\epsilon}_s)\right\|^2\\ &=(T+K)\mathcal{L}_{sa}.\end{aligned}$$

Similarly, by using Jensen's inequality, we can show that

$$\begin{aligned}&(T+K)\mathcal{L}_{sa}\\ &= \sum_{t=1-K}^{T} \mathbf{E}_{\boldsymbol{x}_0,\boldsymbol{\epsilon}_{t:t+K-1}} \left\|\frac{1}{K}\sum_{s=t}^{t+K-1} \tau_s(\boldsymbol{f}_\theta(\boldsymbol{x}_s, s) - \boldsymbol{\epsilon}_s)\right\|^2\\ &\geq \frac{1}{T+K}\mathbf{E}_{\boldsymbol{x}_0,\boldsymbol{\epsilon}} \left\|\sum_{s=2}^{T} \tau_s(\boldsymbol{f}_\theta(\boldsymbol{x}_s, s) - \boldsymbol{\epsilon}_s)\right\|^2\\ &=\frac{1}{T+K}\mathcal{L}_\theta\end{aligned}$$

completing the proof. ⊡

This theorem provides a comparison between our loss and $\mathcal{L}^\tau_{simple}$ which is the weighted conventional loss. Since constants τ_i naturally come from the model formulation and the commonly used loss $\mathcal{L}_{simple}$ ignores those constants, we use the weighted loss for a fair comparison. By using similar arguments with the above proof, it is easy to show that our loss is still tighter than $\mathcal{L}_{simple}$ even when setting every $\tau_i = 1$. This holds for any $K > 1$.

Figure 2: Qualitative results of CelebA-HQ 256 × 256.

Experiments

Image Generation

Experimental setup: In this experiment, we apply the proposed loss to the vanilla DPM, referred to as SA-K-DPM, where K denotes the number of consecutive steps. We evaluate the SA-2-DPM (which we will call SA-DPM for brevity) both individually and in combination with covariance estimation methods, including Analytic-DPM (Bao et al. 2022b), NPR-DPM and SN-DPM (Bao et al. 2022a). All settings and hyperparameters are kept unchanged from (Song, Meng, and Ermon 2021). In particular, the experiments are conducted on: CIFAR10 32×32 (Krizhevsky 2012), CelebA 64×64 (Liu et al. 2015) and one higher-resolution dataset CelebA-HQ 256×256 (Karras et al. 2018). For CIFAR10, the models are trained with two different forward noise schedules: the linear schedule (LS) (Ho, Jain, and Abbeel 2020) and the cosine schedule (CS) (Nichol and Dhariwal 2021). The sampling timesteps for all the datasets are set to $\{10, 50, 100, 200, 1000\}$. For the evaluation, we compute the FID between $50k$ generated images and the pre-computed statistics of the datasets. See more details in Appendix C.1.

Performance Comparison: The summary of sampling performance for CIFAR10 and CelebA is presented in Table 2 and 3. Table 1 presents the results for the remaining dataset CelebA-HQ. Evidently, SA-DPM exhibits a substantial performance improvement over the original DPM, regardless of whether the number of timesteps is small or large. With a large number of timesteps, the original DPM can fully leverage gradient guidance from the denoising model across finer sampling iterations to generate higher-quality samples. However, as the number of timesteps is reduced from 1000 down to 10, the performance gains of our SA-DPM become more pronounced. As observed from those tables, for many settings, 50 or 100 timesteps are sufficient for our method to achieve a similar FID level with prior methods which use 1000 timesteps. This suggests a significant advantage of our new loss to improve both training and

(a) CIFAR10 32 × 32 (b) CelebA 64 × 64

Figure 3: Qualitative results of (a) CIFAR10 32 × 32. (b) CelebA 64 × 64.

Dataset	**# timesteps T**	**Method**							
		B	SA	B+A	SA+A	B+NPR	SA+NPR	B+SN	SA+SN
CIFAR10 32×32 DDPM (LS)	10	41.41	**30.51**	34.19	**21.66**	32.35	**21.10**	24.06	**19.53**
	50	15.98	**9.24**	7.20	**4.20**	6.18	**3.90**	4.63	**3.61**
	100	11.79	**6.73**	5.31	**3.43**	4.52	**3.25**	3.67	**3.10**
	200	9.15	**5.47**	3.92	**3.28**	3.57	**3.16**	3.31	**3.06**
	1000	5.92	**4.33**	3.98	**3.72**	4.10	**3.84**	3.65	**3.56**
CIFAR10 32×32 DDPM (CS)	10	34.98	**24.59**	23.41	**16.66**	19.94	**14.77**	**16.33**	17.23
	50	11.05	**6.27**	5.42	**3.78**	5.31	**3.67**	4.17	**3.97**
	100	8.25	**4.98**	4.45	**3.53**	4.52	**3.51**	3.83	**3.64**
	200	6.69	**4.40**	4.04	**3.53**	4.10	**3.54**	3.72	**3.61**
	1000	4.95	**4.05**	4.26	**3.84**	4.27	**3.87**	4.07	**3.83**
CelebA 64×64 DDPM	10	36.69	**32.15**	28.99	**27.08**	28.37	**26.73**	**20.60**	26.22
	50	18.96	**17.59**	11.23	**9.43**	10.89	**9.42**	7.88	**7.01**
	100	14.31	**12.77**	8.08	**6.53**	8.23	**6.84**	5.89	**5.18**
	200	10.48	**9.14**	6.51	**5.02**	7.03	**5.49**	5.02	**4.04**
	1000	5.95	**4.69**	5.21	**3.99**	5.33	**4.00**	4.42	**3.56**
CelebA 64×64 DDIM	10	20.54	**12.88**	15.62	**10.52**	14.98	**10.48**	**10.20**	19.29
	50	9.33	**7.01**	6.13	**4.18**	6.04	**4.25**	3.83	**3.19**
	100	6.60	**4.81**	4.29	**3.02**	4.27	**3.13**	3.04	**2.62**
	200	4.96	**3.69**	3.46	**2.61**	3.59	**2.76**	2.85	**2.49**
	1000	3.40	**2.98**	3.13	**2.74**	3.15	**2.78**	2.90	**2.66**

Table 2: FID score (↓). The results are reported under different numbers of timesteps T. Here B and SA denote the baseline and our proposed method. A, NPR, and SN denote Analytic-DPM, NPR-DPM, and SN-DPM, respectively.

Dataset	**# timesteps T**	**Method**							
		B	SA	B+A	SA+A	B+NPR	SA+NPR	B+SN	SA+SN
CIFAR10 32×32 DDPM (LS)	10	6.93	**7.55**	8.05	**8.50**	8.17	**8.53**	8.10	**8.42**
	50	8.34	**8.82**	9.53	**9.62**	9.51	**9.63**	9.49	**9.65**
	100	8.59	**9.04**	9.59	**9.74**	9.55	**9.70**	9.47	**9.73**
	200	8.81	**9.15**	9.59	**9.72**	9.49	**9.62**	9.50	**9.65**
	1000	9.03	**9.24**	9.17	**9.37**	9.18	**9.35**	9.24	**9.41**
CIFAR10 32×32 DDPM (CS)	10	7.48	**7.97**	8.05	**8.37**	8.21	**8.49**	8.47	**8.48**
	50	8.53	**9.09**	8.97	**9.43**	9.02	**9.45**	9.10	**9.46**
	100	8.71	**9.20**	9.07	**9.52**	9.09	**9.53**	9.16	**9.54**
	200	8.84	**9.31**	9.14	**9.55**	9.15	**9.54**	9.18	**9.54**
	1000	8.94	**9.45**	9.04	**9.52**	9.04	**9.52**	9.06	**9.54**

Table 3: IS metric (↑). The results are reported under different numbers of timesteps T. Here B and SA denote the baseline and our proposed method. A, NPR, and SN denote Analytic-DPM, NPR-DPM, and SN-DPM, respectively.

Method	λ	# timesteps T				
		10	50	100	200	1000
DDPM	0	41.41	15.98	11.79	9.15	5.92
SA-2-DPM	0.5	35.39	12.09	8.52	6.56	5.25
	1.0	30.51	9.24	6.73	5.47	4.33
	2.0	19.14	10.59	11.21	12.34	14.20
SA-3-DPM	0.3	30.49	10.27	7.63	6.44	5.47
	0.6	23.71	9.07	7.96	7.77	8.06
	1.5	15.59	11.76	13.90	16.34	19.49
SA-4-DPM	0.2	32.93	10.78	7.78	6.17	4.73
	0.4	26.68	9.33	7.53	7.00	6.95

Table 4: FID of CIFAR10 dataset under different weight λ of $\mathcal{L}_{sa}$. We use the sampling type of DDPM to synthesize.

inference in DPMs. For qualitative results, we provide the generated samples of our SA-DPM in Figure 2 and 3.

In addition, we also combine our proposed loss with the three covariance estimation methods (Analytic-DPM, NPR-DPM, and SN-DPM) on two datasets: CIFAR10 and CelebA. Table 2 and 3 show that our loss can boost significantly the image quality. This could be attributed to the capability of our loss to enhance the estimation of the mean of the backward Gaussian distributions in the sampling procedure. So when incorporating the additional covariance estimation methods, the generated image quality is further improved. We further provide synthesized samples in Appendix C.3.

Ablation Study on the Weight λ

In the previous subsection, we used the SA-2-DPM with the weight λ of $\mathcal{L}_{sa}$ set to 1, which resulted in substantial performance improvements when considering small sampling timesteps as compared to the original DPM. Next, we consider the variations in FID scores for CIFAR10 dataset across different configurations of weight $\lambda \in \{0.5, 1, 2\}$ for SA-2-DPM, $\lambda \in \{0.3, 0.6, 1.5\}$ for SA-3-DPM and $\lambda \in \{0.2, 0.4\}$ for SA-4-DPM. In this experiment, the sampling type of DDPM is used for evaluation. As presented in Table 4, all the tested SA-K-DPM methods yield better results compared to the vanilla DPM. With different numbers of consecutive steps, the weight λ plays a crucial role. Specifically, SA-2-DPM ($\lambda = 1$), SA-3-DPM ($\lambda = 0.3$), and SA-4-DPM ($\lambda = 0.2$) consistently outperform DPM for all numbers of sampling timesteps. However, when the weight λ is set much higher, the quality of generated images will degrade slightly when using a large number of timesteps (e.g., 1000), even though it will be significantly better when using a small number of timesteps.

Evaluation on the Estimation Gap

In this experiment, we evaluate the total gap term $\bar{d}_{\theta,t}$ of each trained model during sampling. Because $\bar{d}_{\theta,t}$ contains the weighted sum of the difference between the noise target $\boldsymbol{f}_\theta(\boldsymbol{x}_t, t)$ and the actual noise $\boldsymbol{\epsilon}_t$, however, during the sampling process starting from Gaussian noise $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we cannot know the actual noise due to the unknown input image $\boldsymbol{x}_0$. Therefore, to assess the quantity $\bar{d}_{\theta,t}$ effectively, we take around 2000 input images from the dataset and add noise to them up to time $t = 300$ in order to avoid completely destroying $\boldsymbol{x}_0$. Then, these images $\boldsymbol{x}_{300}$ are used as starting points for the denoising process. At each time step t, we calculate the noise target using the formula $\boldsymbol{\epsilon}_t = \frac{\boldsymbol{x}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0}{\sqrt{1-\bar{\alpha}_t}}$, and then we can compute the gap $\bar{d}_{\theta,t}$.

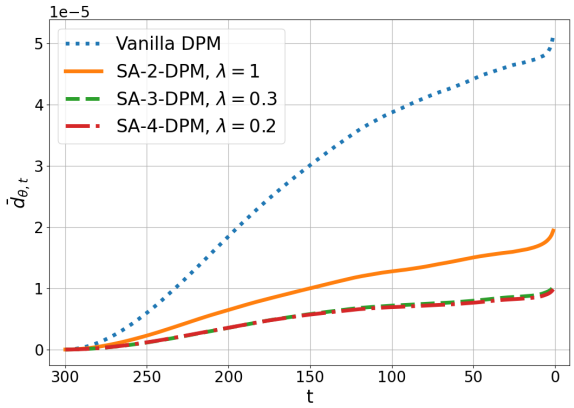


Figure 4: Total gap term $\bar{d}_{\theta,t}$ when sampling image starting from $\boldsymbol{x}_{300}$ on CIFAR10 dataset.

Figure 4 illustrates $\bar{d}_{\theta,t}$ of the sampling process of four trained models on CIFAR10 dataset: vanilla DPM, SA-2-DPM, SA-3-DPM and SA-4-DPM. It can be observed that when training with more consecutive timesteps K in $\mathcal{L}_{sa}$, the total gap term is more effectively minimized during the sampling process. Specifically, with SA-2-DPM, at the final timestep of the denoising process, the total gap term is reduced by approximately 2.5 times compared to the base model. We provide more results in Appendix C.2.

Conclusion

In this work, we examine the estimation gap between the ground truth and predicted trajectory in the sampling process of DPMs. We then propose a sequence-aware loss, that optimizes multiple timesteps jointly to leverage their sequential relationship. We theoretically prove that our proposed loss is a tighter upper bound of the estimation gap than the vanilla loss. Our experimental results verify that our loss reduces the estimation gap and enhances the sample quality. Moreover, when combining our loss with advanced techniques, we achieve a significant improvement over the baselines. Therefore, with our new loss, we provide a new benchmark for future research on DPMs. This new loss represents the true loss of a sampling step and therefore may facilitate future deeper understandings of DPMs, such as generalization ability and optimality. One limitation of this work is that our new loss requires the calculation of the network's output at many timesteps, which makes the training time longer compared to the vanilla loss.

Acknowledgements

This research was partly funded by Vingroup Innovation Foundation (VINIF) under project code VINIF.2022.DA00183.

References

Bao, F.; Li, C.; Sun, J.; Zhu, J.; and Zhang, B. 2022a. Estimating the Optimal Covariance with Imperfect Mean in Diffusion Probabilistic Models. In *International Conference on Machine Learning*, 1555–1584. PMLR.

Bao, F.; Li, C.; Zhu, J.; and Zhang, B. 2022b. Analytic-DPM: an Analytic Estimate of the Optimal Reverse Variance in Diffusion Probabilistic Models. In *International Conference on Learning Representations*.

Chen, N.; Zhang, Y.; Zen, H.; Weiss, R. J.; Norouzi, M.; and Chan, W. 2021. WaveGrad: Estimating Gradients for Waveform Generation. In *International Conference on Learning Representations*.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2017. Density estimation using Real NVP. In *International Conference on Learning Representations*.

Germain, M.; Gregor, K.; Murray, I.; and Larochelle, H. 2015. MADE: Masked Autoencoder for Distribution Estimation. In Bach, F.; and Blei, D., eds., *International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 881–889. Lille, France: PMLR.

Ho, J.; Chen, X.; Srinivas, A.; Duan, Y.; and Abbeel, P. 2019. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, 2722–2730. PMLR.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.

Karras, T.; Aila, T.; Laine, S.; and Lehtinen, J. 2018. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *International Conference on Learning Representations*.

Karras, T.; Aittala, M.; Aila, T.; and Laine, S. 2022. Elucidating the Design Space of Diffusion-Based Generative Models. In *Advances in Neural Information Processing Systems*.

Kingma, D. P.; and Dhariwal, P. 2018. Glow: Generative flow with invertible 1x1 convolutions. *Advances in Neural Information Processing Systems*, 31.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kong, Z.; and Ping, W. 2021. On Fast Sampling of Diffusion Probabilistic Models. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*.

Krizhevsky, A. 2012. Learning Multiple Layers of Features from Tiny Images. *University of Toronto*.

Liu, L.; Ren, Y.; Lin, Z.; and Zhao, Z. 2022. Pseudo Numerical Methods for Diffusion Models on Manifolds. In *International Conference on Learning Representations*.

Liu, Z.; Luo, P.; Wang, X.; and Tang, X. 2015. Deep Learning Face Attributes in the Wild. In *International Conference on Computer Vision (ICCV)*.

Nichol, A. Q.; and Dhariwal, P. 2021. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, 8162–8171. PMLR.

Papamakarios, G.; Pavlakou, T.; and Murray, I. 2017. Masked autoregressive flow for density estimation. *Advances in Neural Information Processing Systems*, 30.

Rezende, D. J.; Mohamed, S.; and Wierstra, D. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, 1278–1286. PMLR.

Salimans, T.; and Ho, J. 2022. Progressive Distillation for Fast Sampling of Diffusion Models. In *International Conference on Learning Representations*.

Sohl-Dickstein, J.; Weiss, E.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, 2256–2265. PMLR.

Song, J.; Meng, C.; and Ermon, S. 2021. Denoising Diffusion Implicit Models. In *International Conference on Learning Representations*.

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

Song, Y.; and Ermon, S. 2019. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.

Song, Y.; Sohl-Dickstein, J.; Kingma, D. P.; Kumar, A.; Ermon, S.; and Poole, B. 2021. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*.

Van den Oord, A.; Kalchbrenner, N.; Espeholt, L.; Vinyals, O.; Graves, A.; et al. 2016. Conditional image generation with pixelcnn decoders. *Advances in Neural Information Processing Systems*, 29.

Watson, D.; Chan, W.; Ho, J.; and Norouzi, M. 2022. Learning Fast Samplers for Diffusion Models by Differentiating Through Sample Quality. In *International Conference on Learning Representations*.

Watson, D.; Ho, J.; Norouzi, M.; and Chan, W. 2021. Learning to efficiently sample from diffusion probabilistic models. *arXiv preprint arXiv:2106.03802*.

Zhang, G.; Niwa, K.; and Kleijn, W. B. 2023. Lookahead Diffusion Probabilistic Models for Refining Mean Estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1421–1429.