SPI-GAN: DENOISING DIFFUSION GANS WITH STRAIGHT-PATH INTERPOLATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

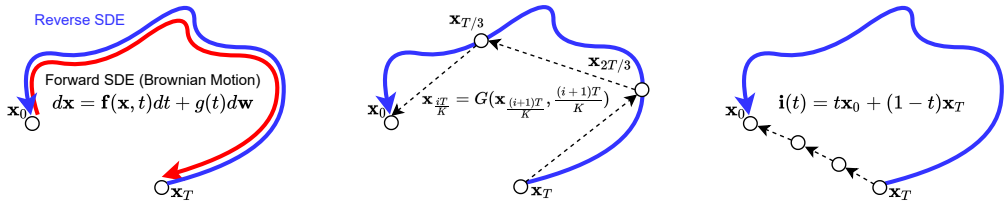
Score-based generative models (SGMs) are a recently proposed paradigm for deep generative tasks and now show the state-of-the-art sampling performance. The original SGM design is known to solve two of the three problems of the generative trilemma except sampling complexity: sampling quality and sampling diversity. That said, their training/sampling complexity is notoriously high. To this end, combining SGMs with simpler models, e.g., generative adversarial networks (GANs), is gathering much attention currently. We present an enhanced denoising method using GANs, called straight-path interpolation GAN (SPI-GAN), which drastically reduces the sampling time while achieving as high sampling quality and diversity as SGMs. Our SPI-GAN can be compared to the state-of-the-art shortcut-based denoising method using GANs, called denoising diffusion GAN (DD-GAN). However, our method corresponds to an extreme method that does not use any intermediate shortcut information of the reverse SDE path, in which case DD-GAN fails to obtain good results (cf. Sec. 4.4). Nevertheless, our straight-path interpolation method greatly stabilizes the overall training process. As a result, SPI-GAN is one of the best-balanced models in terms of the sampling quality/diversity/time for CIFAR-10, CelebA-HQ-256, and LSUN-Church-256.

1 INTRODUCTION

Generative models are one of the most popular research topics for deep learning. Many different models have been proposed, ranging from variational autoencoders (Kingma & Welling, 2013) and generative adversarial networks (Goodfellow et al., 2014) to recent denoising diffusion models (Ho et al., 2020; Song & Ermon, 2019; Song et al., 2021c). The representative denoising diffusion models score matching with Langevin dynamics (Song & Ermon, 2019) and denoising diffusion probabilistic modeling (Ho et al., 2020) progressively corrupt original data and revert the corruption process to build a generative model. Recently, Song et al. (2021c) proposed a stochastic differential equation (SDE)-based mechanism that embraces all those models and coined the term, *score-based generative models* (SGMs).

Each generative model has different characteristics in terms of the generative task trilemma: i) sampling quality, ii) sampling diversity, and iii) sampling time. Generative adversarial networks (GANs) generate samples with high quality, but lack sampling diversity. Conversely, variational autoencoders (VAEs) generate a variety of samples, but its sampling quality is lacking. SGMs outperform GANs and VAEs in terms of sampling quality/diversity. However, it takes a lot more time for sampling. In this paper, we are interested in achieving high sampling quality, diversity, and time simultaneously.

Figure 1 (a) shows the conceptual workflow of SGMs. The corruption process can be described by the forward SDE process with the Wiener process and the reverse SDE with a score function can be recognized as a generative model. It currently shows state-of-the-art quality in synthesizing fake samples for various image datasets. However, one downside of this approach is long sampling time. To this end, some other variations to reduce the computational overheads have been recently proposed (Song & Ermon, 2020; San-Roman et al., 2021; Kong & Ping, 2021; Jolicoeur-Martineau et al., 2021a; Luhman & Luhman, 2021; Xiao et al., 2021b). Figure 1 (b) shows the key idea of one recent method that is the most similar to our research. *Denoising diffusion GAN* (Xiao et al., 2021b) proposed to approximate the reverse SDE process with K shortcuts. They internally utilize a



(a) The SDE-based workflow of SGMs. The reverse SDE is a generation process.

(b) Approximating the reverse SDE (T steps in total) with K shortcuts by DD-GAN, e.g., $K = 3$ in this example.

(c) Our proposed denoising method, SPI-GAN, with K straight-path interpolations, e.g., $K = 3$ in this example.

Figure 1: The comparison among three methods: i) the original formulation of SGMs in (a), ii) DD-GAN's learning the shortcuts of the reverse SDE (based on the conditional generator $\mathbf{x}_{\frac{iT}{K}} = G(\mathbf{x}_{\frac{(i+1)T}{K}}, \frac{(i+1)T}{K})$ in (b), and iii) SPI-GAN's learning the straight path in (c). Note that we do not strictly follow the reverse SDE path but the straightly interpolated path. In the perspective of generative task, the final goal is generate $\mathbf{x}_0$ regardless of the intermediate path. Therefore, one can consider that SPI-GAN denoises $\mathbf{x}_T$ to $\mathbf{x}_0$ following the straight path. One more advantage is that the straight path is much easier to be learned than the highly non-linear reverse SDE path (see Section 4.5 for more detailed discussion).

GAN-based framework conditioned on time (step) t to learn the shortcuts. In contrast to Denoising diffusion GAN (DD-GAN), we propose a denoising method to approximate the straight-path interpolation (guided by intermediate samples by $t\mathbf{x}_0 + (1-t)\mathbf{x}_T$, where $0 \leq t \leq 1$). Our method is called *straight-path interpolation GAN* (SPI-GAN).

One may consider that SPI-GAN is similar to DD-GAN where $K = 1$ which directly denoises x_T to x_0 (cf. Figure 1 (b) vs. (c)). However, our SPI-GAN is technically different and more sophisticated in the following points:

1. Whereas DD-GAN uses the shortcuts following the SDE path for denoising diffusion, SPI-GAN uses the straight path between $\mathbf{x}_T$ and $\mathbf{x}_0$.
2. In order to learn from the interpolated samples, we propose i) a special neural network architecture, characterized by a mapping network, and ii) its own training mechanism. DD-GAN does not have a structure equivalent to our mapping network.
3. SPI-GAN is trained to iteratively denoise $\mathbf{x}_T$ to $\mathbf{x}_0$ following the straight path. For sampling, however, SPI-GAN is able to generate fake images directly without any recursion, which is possible since our mapping network is designed for this purpose (see Sections 3.6 and 4.5).
4. After all these efforts, SPI-GAN produces better and faster outputs in comparison with not only existing denoising models, such as DD-GAN, but also some other generative models.

Both DD-GAN and SPI-GAN can be understood as denoising diffusion model using GANs. Our proposed SPI-GAN shows the best balance in the overall sampling quality, diversity, and time in three benchmark datasets: CIFAR-10, CelebA-HQ-256, and LSUN-Church-256.

2 Related Work and Preliminaries

Neural ordinary differential equations (NODEs). Neural ordinary differential equations (Chen et al., 2018) use the following equation to define the continuous evolving process of the hidden vector $\mathbf{h}(t)$:

$$\mathbf{h}(T) = \mathbf{h}(0) + \int_0^T f(\mathbf{h}(t), t; \boldsymbol{\theta}_\mathbf{f})dt, \tag{1}$$

where the neural network $f(\mathbf{h}(t), t; \boldsymbol{\theta}_\mathbf{f})$ learns $\frac{d\mathbf{h}(t)}{dt}$. To solve the integral problem, we typically rely on various ODE solvers. The explicit Euler method is one of the simplest ODE solvers. The 4th order

Runge–Kutta (RK4) method is a more sophisticated ODE solver and the Dormand–Prince (Dormand & Prince, 1980) method is an advanced adaptive step-size solver. The NODE-based continuous-time models (Chen et al., 2018; Kidger et al., 2020; Brouwer et al., 2019) show good performance in processing sequential data.

Score-based generative models. In diffusion models, the diffusion process is adding noise to real image data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ in T steps as follows:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t\geq 1} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{2}$$

where β_t is a pre-defined variance schedule and $q(\mathbf{x}_0)$ is a data-generating distribution. The denoising (reverse) process of diffusion models is as follow:

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t\geq 1} p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t), \quad p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \sigma_t^2\mathbf{I}), \tag{3}$$

which $\boldsymbol{\theta}$ is denoising model's parameter, and $\mu_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ and σ_t^2 are the mean and variance for the denoising model. Afterwards, score-based generative models (SGMs) generalize the diffusion process to continuous using SDE. SGMs use the following Itô SDE to define diffusive processes:

$$d\mathbf{x} = f(\mathbf{x}, t)dt + g(t)d\mathbf{w}, \tag{4}$$

where $\mathbf{w}$ is the standard Wiener process (a.k.a, Brownian motion), $f(\mathbf{x}, t)$ and $g(t)$ are defined in Appendix A. Following the Equation 4, we can derive an $\mathbf{x}_t$ at time $t \in [0, T]$. As the value of t increases, $\mathbf{x}_t$ approaches to $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. The denoising process (reverse SDE) of SGMs is as follows:

$$d\mathbf{x} = [f(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})]dt + g(t)d\mathbf{w}, \tag{5}$$

where $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is the gradient of the log probability density function. In denoisng process (reverse SDE), a noisy sample at $t = T$ maps to a data sample at $t = 0$.

Compared to other generative models, SGMs generate a variety of images with good quality. However, it takes a lot of time to generate the images. The reason is that a large T (e.g., CIFAR-10: $T = 1000$) is required to approximate $\mathbf{x}_T$ with a Gaussian distribution, and the image is generated through a large number of iterative time steps. To overcome the slow sampling speed with large T, several methods have been proposed including learning an adaptive noise schedule (San-Roman et al., 2021), introducing non-Markovian diffusion processes (Song & Ermon, 2020; Kong & Ping, 2021), using faster SDE solvers for continuous-time models (Jolicoeur-Martineau et al., 2021a), knowledge distillation (Luhman & Luhman, 2021; Salimans & Ho, 2022), and denoising images with GANs (Xiao et al., 2021b; Zheng et al., 2022; Wang et al., 2022).

Among those methods, denoising samples with GANs, e.g., DD-GAN, shows a fast sampling time with minimum quality loss. It effectively reduces the denoising process steps using a GAN. To this end, DD-GAN uses a conditional generator, and $\mathbf{x}_t$ is used as a condition to generate $\mathbf{x}_{t-1}$. It repeats this K times to finally create $\mathbf{x}_0$. For its adversarial learning, the conditional GAN generator can match $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)$ and $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$.

3 Proposed Method

Our proposed method, SPI-GAN, learns, during its training phase, how to denoise $\mathbf{x}_T$ to $\mathbf{x}_0$ following the straight path and after being trained, a latent vector $\mathbf{z}$ is denoised to a fake image *directly without any recursion* (see Sections 3.6 and 4.5).

3.1 Overall Workflow

We first describe the overall workflow of our proposed method. Before describing it, the notations in this paper are defined as follows: i) $\mathbf{i}(t) \in \mathbb{R}^{C\times H\times W}$ is an image with a channel C, a height H, and a width W at interpolation point t. ii) $\hat{\mathbf{i}}(t) \in \mathbb{R}^{C\times H\times W}$ is a generated fake image at interpolation point t. iii) $\mathbf{h}(t) \in \mathbb{R}^{\dim(\mathbf{h})}$ is a latent vector at interpolation point t. iv) $\mathbf{r}(\mathbf{h}(t), t; \boldsymbol{\theta}_{\mathbf{r}}) \in \mathbb{R}^{\dim(\mathbf{h})}$ is a neural network approximating the time derivative of $\mathbf{h}(t)$, denoted $\frac{d\mathbf{h}(t)}{dt}$.

Our proposed SPI-GAN consists of four parts, each of which has a different color in Figure 2, as follows:

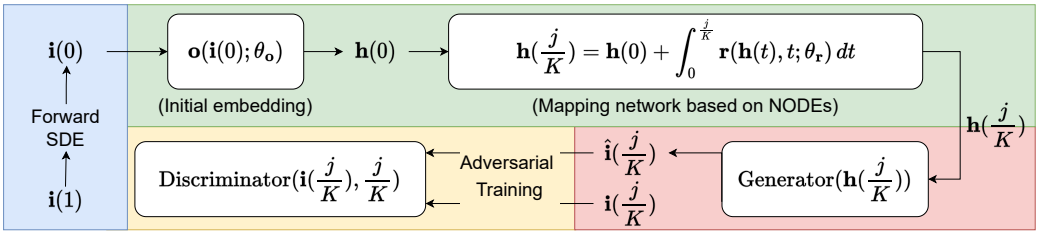


Figure 2: The architecture of our proposed SPI-GAN. We perform this adversarial training for $j \in (0, K]$. $\mathbf{i}(t)$ is the interpolated image at t, and therefore, $\mathbf{i}(1)$ is original image and $\mathbf{i}(0)$ is noisy image. We note that $\mathbf{i}(0)$ theoretically follows a certain Gaussian distribution $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, where σ^2 depends on the type of SDE, and therefore, for generating fake images we do not need $\mathbf{i}(0)$ from real images but $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. Each color represents a computation step (cf. Section 3.1).

1. **1st part (blue):** The first part, highlighted in blue in Figure 2, means that we calculate a noisy image $\mathbf{i}(0) = \mathbf{x}_T$ from $\mathbf{i}(1) = \mathbf{x}_0$ with the forward pass of the SGM. We note that this can be done in $\mathcal{O}(1)$.
2. **2nd part (green):** The second part maps a noisy vector $\mathbf{i}(0)$ into another latent vector $\mathbf{h}(t)$ after solving the integral problem. We note that t can be in $(0, 1]$ to linearly interpolate $\mathbf{i}(0)$ and $\mathbf{i}(1)$. The final integral time of the NODE layer, denoted $\frac{j}{K}$, determines which latent vector it will generate.
3. **3rd part (red):** The third part is a generative step to generate a fake image $\hat{\mathbf{i}}(t)$ from $\mathbf{h}(t)$. Our generator does not require t as input, which means that $\mathbf{h}(t)$ internally has the temporal information. In other words, the latent space, where we sample $\mathbf{h}(t)$, is a common space across $t \in (0, 1]$.
4. **4th part (yellow):** The fourth part is a discriminative step to distinguish between real and fake images. Our discriminator discriminates not only clean images, but also interpolated images for every t. In other words, we maintain only one discriminator regardless of t.
5. After trained for various $t \in (0, 1]$, our generator is able to generate fake images directly without any recursion aided by the mapping network.

3.2 Diffusion through the forward SDE

Depending on the type of SDE, σ^2 can be different for $\mathbf{i}(0) \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. The type of SDE is in Appendix A. Unlike the reverse SDE, which requires step-by-step computation, the forward SDE can be calculated with one-time computation for a target time t (Song et al., 2021c).

3.3 Mapping network

The mapping network, which generates a latent vector $\mathbf{h}(t)$, is the most important component in our model. The mapping network consists of an initial embedding network, denoted $\mathbf{o}$, that generates the initial hidden representation $\mathbf{h}(0)$ from $\mathbf{i}(0)$ and a NODE-based mapping network. The role of network $\mathbf{o}$ is to reduce the size of the input to the mapping network for decreasing sampling time. In addtion, the NODE-based mapping network to generate the latent vector for a target interpolation point t, whose initial value problem (IVP) is defined as follows:

$$\mathbf{h}(\frac{j}{K}) = \mathbf{h}(0) + \int_0^{\frac{j}{K}} \mathbf{r}(\mathbf{h}(t), t; \boldsymbol{\theta}_\mathbf{r})dt, \tag{6}$$

where $\frac{d\mathbf{h}(t)}{dt} = \mathbf{r}(\mathbf{h}(t), t; \boldsymbol{\theta}_\mathbf{r})$, and $\mathbf{r}$ has multiple fully-connected layers in our implementation. In general, $\mathbf{h}(0)$ is a lower-dimensional representation of the input.

One more important point is that we maintain a single latent space for all t and therefore, $\mathbf{h}(t)$ has the information of the image to generate at a target interpolation time t. For instance, Figure 3 shows that a noisy image is generated from $\mathbf{h}(0)$ but a clean image from $\mathbf{h}(1)$. In Figure 4, we also compare the reverse SDE path and the straight interpolation path. In general, the straight interpolation path more quickly converts a noisy image to its corresponding clear image than the reverse SDE path.

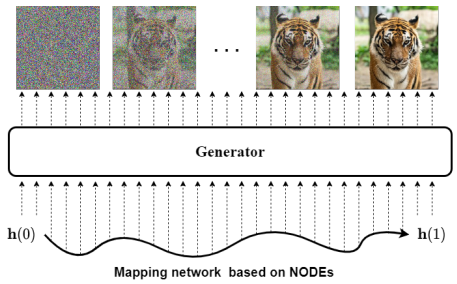


Figure 3: Process of generating sample from continuous latent vector.

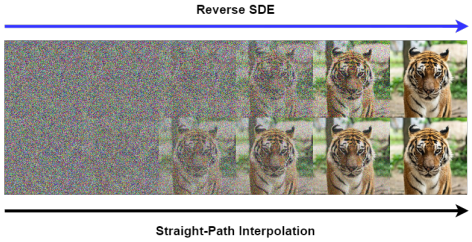


Figure 4: Difference between reverse SDE and interpolation.

3.4 Generative Adversarial Network

SPI-GAN has advantages over traditional GANs. Since traditional GANs only generate clean fake images from noisy vectors, the overfitting issue occurs in the discriminator, which frequently results in mode-collapse at the end of the training process. However, our model is insensitive to the overfitting issue by generating a set of images following the straight interpolation path. The straight-path interpolation (SPI) method is defined as follows:

$$\mathbf{i}(t) = t\mathbf{x}_0 + (1-t)\mathbf{x}_T, \tag{7}$$

where $t = \frac{j}{K}$ and $j \in (0, K]$, and therefore, $\mathbf{i}(1) = \mathbf{x}_0$ and $\mathbf{i}(0) = \mathbf{x}_T$. If we use $K = 3$, there are 3 interpolated images $\{\mathbf{i}(t_k)\}_{k=1}^{3}$, where $t_k \in (0, 1]$, $t_{k-1} < t_k$, and $t_3 = 1$, including the original image $\mathbf{i}(1)$. During our training process, we randomly sample t_k rather than using a fixed interval of $\frac{1}{K}$ between t_{k-1} and t_k, which increases the diversity of interpolated images (cf. Table 8). Our objective function to train our proposed model is in Appendix B.5

3.4.1 Generator

Our generator is similar to that of StyleGAN2 (Karras et al., 2020b). Therefore, we borrow the network structure of StyleGAN2. However, the biggest difference from StyleGAN2 is that our mapping network is a continuous-time model and generates a latent vector at various interpolation points while maintaining one latent space across them. This is the key point in our model design to generate the interpolated image $\hat{\mathbf{i}}(t)$ with various t settings. We refer to Appendix B.4 for the detailed network structure.

3.4.2 Discriminator

The discriminator of SPI-GAN is time-dependent, unlike the discriminator of traditional GANs. Therefore, we denote the discriminator as a mapping function Discriminator: $\mathbb{R}^{C\times H\times W} \times \mathbb{R}^{P} \rightarrow [0, 1]$, where P is the dimension size of the time embedding. The discriminator receives $\hat{\mathbf{i}}(t)$ and the embedding of t as input and learns to classify images from various interpolation points. As a result, it solves the overfitting problem that traditional GANs have since our discriminator sees various clean and noisy images. DD-GAN also uses this strategy to overcome the overfitting problem. As shown in Figure 4, in addition, the straight-path interpolation method maintains a better balance between noisy and clean images than the case where we sample following the reverse SDE path. Therefore, our discriminator learns a more balanced set of noisy and clean images than that of DD-GAN. We refer to Appendix B.4 for the detailed network structure.

3.5 Training Algorithm

Our training algorithm is in Algorithm 1. In each iteration, we first create a mini-batch of N real images, denoted $\{\mathbf{x}_0^l\}_{l=1}^{N}$. Using the forward SDE, we derive a mini-batch of N noisy images, denoted $\{\mathbf{x}_T^l\}_{l=1}^{N}$. We then sample a set of values for $j \in \{j_1, j_2, \cdots, j_K\}$, where $j_k \sim \mathcal{U}(0, K)$,

Algorithm 1: How to train SPI-GAN

Input: Training data D_{train}, Maximum iteration numbers max_iter, Parameter of discriminator ϕ, Parameter of mapping network ψ, Parameter of generator θ

Initialize ϕ, ψ, θ;
iter $\leftarrow 0$;
while iter $<$ max_iter **do**
  Create a mini-batch of real images $\{\mathbf{x}_0^l\}_{l=1}^N$, where $\mathbf{x}_0^l$ means l-th real image;
  Calculate a mini-batch of noisy images $\{\mathbf{x}_T^l\}_{l=1}^N$ with the forward SDE, where $\mathbf{x}_T^l \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$;
  for $j \in \{j_1, j_2, \cdots, j_K\}$, where $j_k \sim \mathcal{U}(0, K)$, $j_{k-1} < j_k$, and $j_K = K$ **do**
    Calculate $\{\mathbf{h}^l(t)\}_{l=1}^N$, where $t = \frac{j}{K}$, with the mapping network which processes $\{\mathbf{x}_T^l\}_{l=1}^N$;
    Generate a fake image $\{\hat{\mathbf{i}}^l(t)\}_{l=1}^N$ with the generator;
    if iter $\mod 2 \equiv 0$ **then**
      Calculate $\{\mathbf{i}^l(t)\}_{l=1}^N$ with eq:interpolation ;
      Update ϕ via adversarial training;
    else
      Update ψ and θ via adversarial training;
  iter $\leftarrow$ iter $+ 1$;
return ϕ, ψ, θ;

$j_{k-1} < j_k$, and $j_K = K$. After that, our mapping network generates a set of latent vectors, denoted $\{\mathbf{h}^l(t)\}_{l=1}^N$. Our generator then produces a set of fake images from the generated latent vectors. After that, we follow the standard adversarial training sequence (cf. Appendix B.5).

Implementation trick. We need to calculate $\mathbf{h}^l(t)$ for all $t = \frac{j}{K}$, where $j \in (0, K]$. However, we do not repeat the computation in the Equation 6 for each j, but incrementally solve the initial value problem from $j-1$ to j. In this way, we can minimize efficiently the overall computation.

3.6 How to generate

In order to generate samples with SPI-GAN, we need only $\mathbf{h}(1)$ from the mapping network. Unlike other auto-regressive denoising models that require K steps when generating samples as in Figure 1 (b), SPI-GAN learns the denoising path using a NODE-based mapping network. After sampling $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, we feed them into the mapping network to derive $\mathbf{h}(1)$ — we solve the initial value problem in the Equation 6 from 0 to 1 — and our generator generates a fake image $\hat{\mathbf{i}}(1)$. In other words, it is possible to generate latent vector $\mathbf{h}(1)$ directly in our case, which is later used to generate a fake sample $\hat{\mathbf{i}}(1)$. (see Appendix B.7 for sampling algorithm)

Table 1: Differences in image generation.

	DD-GAN	SPI-GAN
Train	$\mathbf{x}(\frac{j}{K})$ of the reverse SDE, $\forall j$	$\mathbf{i}(\frac{j}{K})$ of the straight path, $\forall j$
Generate	Recursively generate $\hat{\mathbf{x}}(\frac{j}{K})$, $\forall j$	Directly generate $\hat{\mathbf{i}}(1)$

4 Experiments

We describe our experimental environments and results. In Appendix B, more detailed experimental settings including software/hardware and hyperparameters, for reproducibility. We also release our model with trained checkpoints.

4.1 Experimental environments

Diffusion types. Among various types, we conduct experiments based on the variance preserving SDE (VP-SDE) for their high sampling quality and reliability, which makes $\sigma^2 = 1$. That is, $\mathbf{i}(1)$ and $\mathbf{z}$ follow a unit Gaussian distribution (see Appendix B.2 for more descriptions).

Datasets. We use CIFAR-10 (Krizhevsky et al., 2014), CelebA-HQ-256 (Karras et al., 2018), and LSUN-Church-256 (Yu et al., 2015). CIFAR-10 has a resolution of 32x32 and is one of the most widely used datasets. CelebA-HQ-256 and LSUN-Church-256 contain high-resolution images of 256x256. Each of them has many real-world images.

Table 2: Results of the unconditional generation on CIFAR-10.

Model	IS ↑	FID ↓	Recall↑	NFE↓	Time↓
SPI-GAN (ours), $K = 2$	**10.3**	3.09	**0.66**	1	0.05
Denoising Diffusion GAN (DD-GAN), $K = 4$ (Xiao et al., 2021b)	9.63	3.75	0.57	4	0.21
DDPM (Ho et al., 2020)	9.46	3.21	0.57	1000	80.5
NCSN (Song & Ermon, 2019)	8.87	25.3	-	1000	107.9
Adversarial DSM (Jolicoeur-Martineau et al., 2021b)	-	6.10	-	1000	-
Likelihood SDE (Song et al., 2021b)	-	2.87	-	-	-
Score SDE (VE) (Song et al., 2021c)	9.89	2.20	0.59	2000	423.2
Score SDE (VP) (Song et al., 2021c)	9.68	2.41	0.59	2000	421.5
Probability Flow (VP) (Song et al., 2021c)	9.83	3.08	0.57	140	50.9
LSGM (Vahdat et al., 2021)	9.87	**2.10**	0.61	147	44.5
DDIM, T=50 (Song et al., 2021a)	8.78	4.67	0.53	50	4.01
FastDDPM, T=50 (Kong & Ping, 2021)	8.98	3.41	0.56	50	4.01
Recovery EBM (Gao et al., 2021)	8.30	9.58	-	180	-
Improved DDPM (Nichol & Dhariwal, 2021)	-	2.90	-	4000	-
VDM (Kingma et al., 2021)	-	4.00	-	1000	-
UDM (Kim et al., 2021)	10.1	2.33	-	2000	-
D3PMs (Austin et al., 2021)	8.56	7.34	-	1000	-
Gotta Go Fast (Jolicoeur-Martineau et al., 2021a)	-	2.44	-	180	-
DDPM Distillation (Luhman & Luhman, 2021)	8.36	9.36	0.51	1	-
SNGAN (Miyato et al., 2018)	8.22	21.7	0.44	1	-
SNGAN+DGflow (Ansari et al., 2021)	9.35	9.62	0.48	25	1.98
AutoGAN (Gong et al., 2019)	8.60	12.4	0.46	1	-
TransGAN (Jiang et al., 2021)	9.02	9.26	0.48	25	-
StyleGAN2 w/o ADA (Karras et al., 2020b)	9.18	8.32	0.41	1	0.04
StyleGAN2 w/ ADA (Karras et al., 2020a)	9.83	2.92	0.49	1	0.04
StyleGAN2 w/ Diffaug (Zhao et al., 2020)	9.40	5.79	0.42	1	0.04
Glow (Kingma & Dhariwal, 2018)	3.92	48.9	-	1	-
PicxelCNN (Van Oord et al., 2016)	4.60	65.9	-	1024	-
NVAE (Vahdat & Kautz, 2020)	7.18	23.5	0.51	1	0.36
IGEBM (Du & Mordatch, 2019)	6.02	40.6	-	60	-
VAEBM (Xiao et al., 2021a)	8.43	12.2	0.53	16	8.79

Evaluation metrics. We use 5 evaluation metrics to quantitatively evaluate fake images. The inception score (Salimans et al., 2016) and the Fréchet inception distance (Heusel et al., 2017) are traditional methods to evaluate the fidelity of fake samples. The improved recall (Kynkäänniemi et al., 2019) reflects whether the variation of generated data matches the variation of training data. Finally, the number of function evaluations (NFE) and wall-clock time (Time) are used to evaluate the generation time for a batch size of 100 images.

4.2 Main results

In this subsection, we evaluate our proposed model quantitatively and qualitatively. For CIFAR-10, we perform the unconditional image generation task for fair comparisons with existing models. The quantitative evaluation results are shown in Table 2. Although our Fréchet inception distance (FID) is 0.68 worse than that of the Score SDE (VP), it shows better scores in all other metrics. However, our method has a better FID score than that of DD-GAN, one of the most related methods. DD-GAN is inferior to our method for all those three quality metrics. LGSM also shows high quality for FID. However, its inception score (IS) and improved recall (Recall) scores are worse than ours. Our method's sample generation time (Time) is almost the same as that of StyleGAN2, which is one of the fastest methods. In summary, SPI-GAN not only increases the quality of samples but also decreases the sampling time.

In addition, our model shows outstanding performance in all evaluation metrics compared to DD-GAN. In particular, SPI-GAN, unlike DD-GAN, does not increase the sample generation time even when K is large — in fact, K only affects the training time of our method since for the generation, we always use $j = K$ regardless of how large/small K is (cf. Section 3.6). Therefore, there is no trade-off between K and the generation time, which is one good characteristic of our method.

Even for high-resolution images, our model shows good performance. In particular, our method shows the best FID score for CelebA-HQ-256 in Table 3, which shows the efficacy of our proposed method. However, our method does not produce significant improvements for LSUN-Church-256 in Table 4 — our method outperforms DD-GAN with other metrics in Table 6. The qualitative results are in Figures. 5, 6, and 7. As shown, our method is able to generate visually high-quality images. More detailed images are in Appendix C

Table 3: Results on CelebA-HQ-256.

Model	FID↓
SPI-GAN (ours)	**6.62**
DD-GAN	7.64
Score SDE	7.23
LSGM	7.22
UDM	7.16
NVAE	29.7
VAEBM	20.4
NCP-VAE (Aneja et al., 2021)	24.8
PGGAN (Karras et al., 2018)	8.03
Adv. LA (Pidhorskyi et al., 2020)	19.2
VQ-GAN (Esser et al., 2021b)	10.2
DC-AE (Parmar et al., 2021)	15.8

Table 4: Results on LSUN-Church-256.

Model	FID↓
SPI-GAN (ours)	6.03
DD-GAN	5.25
DDPM	7.89
ImageBART (Esser et al., 2021a)	7.32
Gotta Go Fast	25.6
PGGAN	6.42
StyleGAN (Karras et al., 2019)	4.21
StyleGAN2	3.86
CIPs (Anokhin et al., 2021)	**2.92**

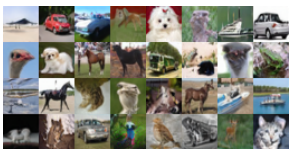


Figure 5: Qualitative results on CIFAR-10.

Figure 6: Qualitative results on CelebA-HQ-256.

Figure 7: Qualitative results on LSUN-Church-256.

4.3 Training time analyses

As clarified earlier, our method does not require training the reverse path of SGMs but its theoretical forward path equation is enough (for running the blue step in Figure 2). Therefore, our overall training consists of training our model only, which takes 6 days with 2 A6000 GPUs for CelebA-HQ-256. For training DD-GAN, it takes 6.5 days with 8 A100 GPUs. Training the original SGM model takes 10 days with 8 A100 GPUs. In comparison with them, our method is faster.

4.4 Ablation studies

In this subsection, we conduct experiments by changing i) the number of interpolations, K, and ii) fixing the intermediate points, $j = \{1, 2, \cdots, K\}$, which are the two most important hyperparameters.

The number of interpolations. According to Table 5, the sampling fidelity, i.e., IS and FID, is the best when $K = 2$, and Recall indicating the sampling diversity is the best when $K = 3$. In general, the higher the value of K is, the higher the sampling diversity is and the lower the sampling fidelity is. When $K = 1$, the discriminator uses only clean images ($\mathbf{i}(1)$) for learning. Comparison with DD-GAN ($K = 1$) shows that our proposed model is more sophisticated than DD-GAN.

Fixed intermediate points. In our model, $j \sim \mathcal{U}(0, K)$ is stochastic between 0 and K. However, we can fix $j \in \{0, 1, \ldots, K\}$. That is, it does not learn about various j, but only train with a specific section (e.g., when $K = 2$, $j \in \{0, 1, 2\}$). It can be seen that in Table 5 and Figure 8, however, our original stochastic setting not only shows better sampling quality than the fixed setting but also converges more rapidly.

Table 5: Ablation studies on CIFAR-10.

Model	IS ↑	FID↓	Recall↑
$K = 1$ (DD-GAN)	8.93	14.6	0.19
$K = 1$ (SPI-GAN)	9.42	10.5	0.60
$K = 2$	**10.3**	**3.09**	0.66
$K = 3$	9.65	8.45	**0.68**
$K = 4$	9.68	5.49	0.67
$K = 5$	9.61	6.25	0.65
Fixed j	8.58	20.8	0.63

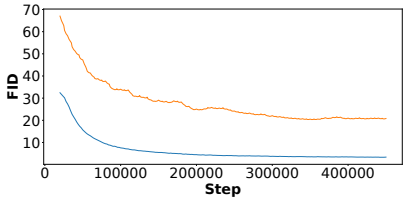


Figure 8: FID curves of the stochastic with $K = 2$ (blue) vs. the fixed j (orange).

4.5 Additional studies

We introduce additional studies for advantage of denoising method with straight-path interpolation and results for evaluating sampling quality and interpolations.

Effectiveness of the straight-path interpolation. Between $\mathbf{x}_T$ and $\mathbf{x}_0$, our straight-path interpolation provides a much simpler path than that of DD-GAN because it follows the linear equation in Equation 7. Also, because of the nature of the linear interpolation, its training is robust, even when $\mathbf{i}(t + \Delta t)$ is missing, if $\mathbf{i}(t)$ and $\mathbf{i}(t + 2\Delta t)$ are considered. This is not guaranteed if a path from $\mathbf{x}_T$ to $\mathbf{x}_0$ is non-linear. As a result, SPI-GAN using the straight-path interpolation shows better performance and robustness with fewer intermediate interpolation points, e.g., our best setting is $K = 2$ vs. DD-GAN uses $K = 4$.

Table 6: Improved quality metrics on LSUN-Church-256.

Model	Recall↑	Coverage↑
SPI-GAN	0.28	**0.65**
DD-GAN	0.16	0.58
CIPs	**0.43**	0.57

Improved metrics. FID is one of the most popular evaluation metrics to measure the similarity between real and generated images. Generated images are sometimes evaluated for the fidelity and diversity using the improved precision and recall (Kynkäänniemi et al., 2019). However, the recall is not accurately detecting similarities between two distributions and is not robust against outliers. Therefore, we evaluate the generated images with the coverage (Naeem et al., 2020) which overcome the limitations of the recall. We compare our method with CIPs, which marks the best FID score in LSUN-Church-256, and the results are in Table 6. Our SPI-GAN outperforms CIPs in terms of the coverage.

Generation by manipulating $\mathbf{z}$ and $\mathbf{h}$. There are three manipulations parts in our model. The first one is generating by changing the latent vector from $\mathbf{h}(0)$ to $\mathbf{h}(1)$ in Figure 9, the second one is the interpolation between two noisy vectors $\mathbf{z}'$ and $\mathbf{z}''$ in Figure 10, and the last one is the interpolation between two latent vector $\mathbf{h}(1)'$ to $\mathbf{h}(1)''$ in Figure 11. In Figure 9, the ideal generation is that noises are gradually removed for an image, but similar images, not the same, are produced. However, the denoising patterns can be observed well. Figure 10 and Figure 11 show the interpolation of the noise vector ($\mathbf{z}$) and the latent vector ($\mathbf{h}(1)$). One can observe that generated images are gradually changed from a mode to another.

Figure 9: Generation by varying the latent vector from $\mathbf{h}(0)$ to $\mathbf{h}(1)$ given fixed $\mathbf{z}$.

Figure 10: Generation by interpolating $\mathbf{z} = (1 - a)\mathbf{z}' + a\mathbf{z}''$, where $0 \leq a \leq 1$.

Figure 11: Generation by interpolating $\mathbf{h}(1) = (1 - a)\mathbf{h}(1)' + a\mathbf{h}(1)''$, where $0 \leq a \leq 1$.

5 Conclusions

Score-based generative models (SGMs) now show the state-of-the-art performance in image generation. However, the sampling time of SGMs is significantly longer than other generative models, such as GANs, VAEs, and so on. Therefore, we presented the most balanced model by reducing the sampling time, called SPI-GAN. Our proposed denoising method uses the straight-path interpolation. Also, it can directly generate fake samples without any recursive computation on time (or step). Our method shows the best sampling quality in various metrics and faster sampling time than other score-based methods. Our ablation and additional studies show the effectiveness of our proposed model. One limitation is that our method fails to achieve the best results for FID in LSUN-Church-256.

6 ETHICS STATEMENT

Generative models are growing rapidly. In particular, score based generative models generate more realistic images. In this situation, our proposed model can generate high-quality images quickly by reducing inference time significantly compared to previous models. Although there are positive aspects to this research direction, there may be negative aspects such as malicious video generation and image synthesis.

7 REPRODUCIBILITY STATEMENT

For reproducibility, we attached the sources codes and trained checkpoints in our supplementary materials. There are detailed descriptions for experimental environment settings, datasets, training processes, evaluation and visualization processes in README.md. In Appendix, we also list all the detailed neural network architectures and their hyperparameters.

REFERENCES

Jyoti Aneja, Alex Schwing, Jan Kautz, and Arash Vahdat. A contrastive learning approach for training variational autoencoder priors. *Advances in Neural Information Processing Systems*, 34, 2021.

Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lempitsky, and Denis Korzhenkov. Image generators with conditionally-independent pixel synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14278–14287, 2021.

Abdul Fatir Ansari, Ming Liang Ang, and Harold Soh. Refining deep generative models via discriminator gradient flow. *arXiv preprint arXiv:2012.00780*, 2021.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. In *NeurIPS*, 2019.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

J.R. Dormand and P.J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19 – 26, 1980.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32, 2019.

Patrick Esser, Robin Rombach, Andreas Blattmann, and Bjorn Ommer. Imagebart: Bidirectional context with multinomial diffusion for autoregressive image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021a.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12873–12883, 2021b.

Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. *arXiv preprint arXiv:2012.08125*, 2021.

Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. Autogan: Neural architecture search for generative adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3224–3234, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two pure transformers can make one strong gan, and that can scale up. *Advances in Neural Information Processing Systems*, 34, 2021.

Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*, 2021a.

Alexia Jolicoeur-Martineau, Rémi Piché-Taillefer, Rémi Tachet des Combes, and Ioannis Mitliagkas. Adversarial score matching and improved sampling for image generation. *arXiv preprint arXiv:2009.05475*, 2021b.

Minguk Kang, Joonghyuk Shin, and Jaesik Park. Studiogan: A taxonomy and benchmark of gans for image synthesis. *arXiv preprint arXiv:2206.09479*, 2022.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2018.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114, 2020a.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8110–8119, 2020b.

Patrick Kidger, James Morrill, James Foster, and Terry J. Lyons. Neural controlled differential equations for irregular time series. In *NeurIPS*, 2020.

Dongjun Kim, Seungjae Shin, Kyungwoo Song, Wanmo Kang, and Il-Chul Moon. Score matching model for unbounded data score. *arXiv preprint arXiv:2106.05527*, 2021.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *arXiv preprint arXiv:2107.00630*, 2021.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

Zhifeng Kong and Wei Ping. On fast sampling of diffusion probabilistic models. *arXiv preprint arXiv:2106.00132*, 2021.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55(5), 2014.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International Conference on Machine Learning*, pp. 7176–7185. PMLR, 2020.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021.

Gaurav Parmar, Dacheng Li, Kwonjoon Lee, and Zhuowen Tu. Dual contradistinctive generative autoencoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 823–832, 2021.

Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14104–14113, 2020.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Robin San-Roman, Eliya Nachmani, and Lior Wolf. Noise estimation for generative diffusion models. *arXiv preprint arXiv:2104.02600*, 2021.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2021a.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 34, 2021b.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2021c.

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679, 2020.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34, 2021.

Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pp. 1747–1756. PMLR, 2016.

Zhendong Wang, Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Diffusion-gan: Training gans with diffusion. *arXiv preprint arXiv:2206.02262*, 2022.

Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. Vaebm: A symbiosis between variational autoencoders and energy-based models. *arXiv preprint arXiv:2010.00654*, 2021a.

Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021b.

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *Advances in Neural Information Processing Systems*, 33:7559–7570, 2020.

Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Truncated diffusion probabilistic models. *arXiv preprint arXiv:2202.09671*, 2022.