APPENDIX

A STOCHASTIC DIFFERENTIAL EQUATION (SDE)

In (Song et al., 2021c), three types of SDE diffusion processes are presented. Depending on the type, $f(\mathbf{x}, t)$ and $g(t)$ are defined as follows:

$$f(\mathbf{x}, t) = \begin{cases} 0, & \text{if VE-SDE,} \\ -\frac{1}{2}\beta(t)\mathbf{x}, & \text{if VP-SDE,} \\ -\frac{1}{2}\beta(t)\mathbf{x}, & \text{if sub-VP-SDE,} \end{cases} \tag{8}$$

$$g(t) = \begin{cases} \sqrt{\frac{\mathrm{d}[\sigma^2(t)]}{\mathrm{d}t}}, & \text{if VE-SDE,} \\ \sqrt{\beta(t)}, & \text{if VP-SDE,} \\ \sqrt{\beta(t)(1 - e^{-2\int_0^t \beta(s)\,\mathrm{d}s})}, & \text{if sub-VP-SDE,} \end{cases} \tag{9}$$

where $\sigma^2(t)$ and $\beta(t)$ are functions w.r.t. time t. Full derivatives of VE, VP and sub-VP SDE are presented in (Song et al., 2021c, Appendix. B).

B 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 the SDE path because it follows the linear equation in Eq. equation 6. Also, because of the nature of the linear interpolation, its training is robust, even when $\mathbf{i}(u + \Delta u)$ is missing, if $\mathbf{i}(u)$ and $\mathbf{i}(u + 2\Delta u)$ 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.

One can also derive the following ordinary differential equation (ODE) from the straight-path interpolation definition in Eq. equation 6 after taking the derivative w.r.t. u:

$$\frac{d\mathbf{i}(u)}{du} = \mathbf{x}_0 - \mathbf{x}_T, \tag{10}$$

where one can get $\mathbf{i}(u + h) = \mathbf{i}(u) + h\frac{d\mathbf{i}(u)}{du} = (u + h)\mathbf{x}_0 + (1 - u - h)\mathbf{x}_T$ with the Euler method. In comparison with the SDE in Eqs. equation 8 and equation 9, the ODE provides a much simpler process.

C EFFECTIVENESS OF NEURAL ORDINARY DIFFERENTIAL EQUATIONS-BASED MAPPING NETWORK

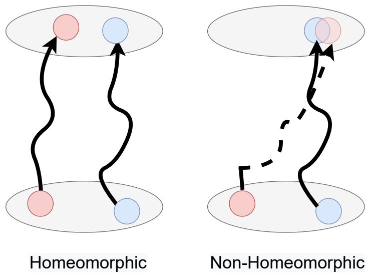

Figure 9: In a homeomorphic mapping, the relative positions of the red and blue particles cannot be switched after the mapping.

As we mentioned in the related work section, NODEs are able to model continuous dynamics of hidden vectors over time using the following method:

$$\mathbf{h}(u) = \mathbf{h}(0) + \int_0^u f(\mathbf{h}(t), t; \boldsymbol{\theta}_\mathbf{f})dt, \tag{11}$$

where the neural network $f(\mathbf{h}(t), t; \boldsymbol{\theta}_\mathbf{f})$ learns $\frac{d\mathbf{h}(t)}{dt}$. To derive $\mathbf{h}(u)$, we solve the integral problem, and in this process, there is one well-known characteristic of NODEs. Let $\psi_t : \mathbb{R}^{\dim(\mathbf{h}(0))} \rightarrow \mathbb{R}^{\dim(\mathbf{h}(u))}$ be a mapping from 0 to u generated by an ODE after solving the integral problem. It is widely known that ψ_t becomes a homeomorphic mapping: ϕ_t is continuous and bijective and ϕ_t^{-1} is also continuous for all $t \in [0, T]$, where T is the last time point of the time domain (Dupont et al., 2019; Massaroli et al., 2020). From this characteristic, the following proposition can be derived: the topology of the input space of ϕ_t is

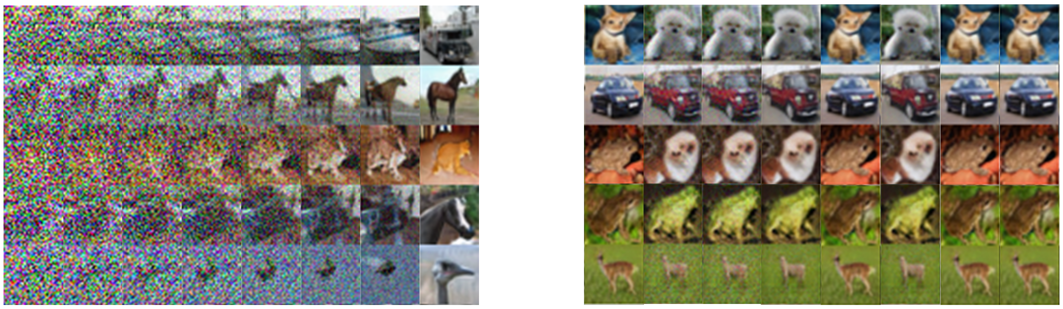

Figure 10: Comparing $\hat{\mathbf{i}}(u)$ according to the mapping network type. Given a fixed z, $\hat{\mathbf{i}}(u)$ generated by varying the latent vector from $\mathbf{h}(0)$ to $\mathbf{h}(1)$ **Left:** our NODE-based mapping network. **Right:** the StyleGAN2's original mapping network which is quickly trained to overlook u

preserved in its output space, and therefore, hidden trajectories crossing each other cannot be represented by NODEs — one can consider that topology as relative positions among particles. Therefore, our NODE-based mapping network can learn the hidden dynamics of $\mathbf{h}(u)$ for all $u \in (0, 1]$ while maintaining the topology of $\mathbf{h}(u)$ at $t = 0$ (e.g., Fig. 9).

Figure 10 shows one advantage of the homeomorphic mapping of SPI-GAN. SPI-GAN with a NODE-based mapping network, shows an appropriate denoising process when $\hat{\mathbf{i}}(0)$ is generated using $\mathbf{h}(0)$ to $\mathbf{h}(1)$, given a fixed z. In contrast, StyleGAN2's mapping network (non-homeomorphic) do not show a denoising process. In other words, NODE-based SPI-GAN with the homeomorphic characteristic can learn the denoising process, but SPI-GAN with the non-homeomorphic mapping network are collapsed into clean images only. We think that this is because i) the time information u is concatenated with the input to the StyleGAN2's original non-homeomorphic mapping network, but ii) it is trained to overlook u. This justifies our design choice to explicitly model the hidden dynamics with the homeomorphic NODE.

D SPI-GAN DETAILS

In this section, we refer to the detailed model architecture, object function, and training algorithm of our proposed SPI-GAN.

D.1 MODEL ARCHITECTURE

The network architectures of StyleGAN2 are modified to implement our proposed straight-path interpolation after adding the NODE-based mapping network and customizing some parts.

Mapping network. Our mapping network consists of two parts. First, the network architecture to define the function $\mathbf{o}$ is in Table 6. Second, the NODE-based network has the following ODE function $\mathbf{r}$ in Table 7.

Table 6: The architecture of the network $\mathbf{o}$.

Layer	Design	Input Size	Output Size
1	LeakyReLU(Linear)	$C \times H \times W$	$\dim(\mathbf{h})$

Table 7: The architecture of the network $\mathbf{r}$.

Layer	Design	Input Size	Output Size
1	LeakyReLU(Linear)	$\dim(\mathbf{h})$	$\dim(\mathbf{h})$

Generator. We follow the original StyleGAN2 architecture. However, we use the latent vector $\mathbf{h}(u)$ instead of the intermediate latent code w of StyleGAN2.

Discriminator. The network architecture of the discriminator is also based on StyleGAN2 (Karras et al., 2020b) — StyleGAN2 has two versions for the discriminator, i.e., Original and Residual. However, our discriminator receives time u as a conditional input. To this end, we use the positional embedding of the time value as in (Ho et al., 2020). The hyperparameters for the discriminator are in Table 8.

D.2 OBJECT FUNCTION

We train our model using the Adam optimizer for training both the generator and the discriminator. We use the exponential moving average (EMA) when training the generator, which achieves high performance in (Ho et al., 2020; Song et al., 2021c; Karras et al., 2020a). The hyperparameters for the optimizer are in Table 8. The adversarial training object of our model is as follows:

$$\begin{aligned}&\min_{\phi} \mathbb{E}_{\mathbf{i}(u)\sim \mathbf{q}_{\mathbf{i}(u)}}\big[-\log(D_\phi(\mathbf{i}(u),u))\\&+\mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})}\big[-\log(1-D_\phi(G_\theta(M_\psi(\mathbf{z})),u))\big]\big],\\&\max_{\theta,\psi}\mathbb{E}_{\mathbf{i}(t)\sim \mathbf{q}_{\mathbf{i}(u)}}\big[\mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})}\big[\log(D_\phi(G_\theta(M_\psi(\mathbf{z})),u))\big]\big],\end{aligned} \tag{12}$$

where, $\mathbf{q}_{\mathbf{i}(u)}$ is the interpolated image distribution, D_ϕ is denoted as the discriminator, G_θ is denoted as the generator, and M_ψ is denoted as the mapping network of our model. We also use the R_1 regularization and the path length regularization (Karras et al., 2020b). λ_{R_1} (resp. λ_{path}) means the coefficient of the R_1 regularization term (resp. the coefficient of the path length regularization term). Each regularizer term is as follows:

$$R_1(\phi) = \lambda_{R_1}\mathbb{E}_{q(\mathbf{i}(u))}\big[\|\nabla_{\mathbf{i}(t)}(D_\phi(\mathbf{i}(u)|u))\ \|^2\big], \tag{13}$$

$$\text{Path length} = \lambda_{path}\mathbb{E}_{\mathbf{h}(u),\mathbf{i}(u)}\big(\|\mathbf{J}^T_{\mathbf{h}(u)}\mathbf{i}(u)\ \|_2 - a\big), \tag{14}$$

where $\mathbf{J}_{\mathbf{h}(u)} = \partial G_\theta(\mathbf{h}(u))/\partial \mathbf{h}(u)$ is the Jacobian matrix. The constant a is set dynamically during optimization to find an appropriate global scale. The path length regularization helps with the mapping from latent vectors to images. The lazy regularization makes training stable by computing the regularization terms (R_1, path length) less frequently than the main loss function. In SPI-GAN, the regularization term for the generator and the discriminator is calculated once every 4 iterations and once every 16 iterations, respectively. The hyperparameters for the regularizers are in Table 8.

D.3 TRAINING ALGORITHM

There is a training algorithm. The two main differences with the sampling algorithm are i) we sample a set of noisy vectors $\{\mathbf{z}^l\}_{l=1}^N$ whereas we use $\{\mathbf{x}_T^l\}_{l=1}^N$ in the training algorithm, and ii) for sampling, we need only $\{\mathbf{h}^l(1)\}_{l=1}^N$.

Algorithm 2 How to train SPI-GAN

```
Input: Training data $D_{\text{train}}$, Maximum iteration numbers $max\_iter$
Initialize discriminator $\phi$, mapping net. $\psi$, generator $\theta$
$iter \leftarrow 0$
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 path
    Sample $u$, where $u \in (0, 1]$
    Calculate $\{\mathbf{h}^l(u)\}_{l=1}^N$ with the mapping network which processes $\{\mathbf{x}_T^l\}_{l=1}^N$
    Generate fake images $\{\hat{\mathbf{i}}^l(u)\}_{l=1}^N$ with the generator
    if $iter \mod 2 \equiv 0$ then
        Calculate $\{\mathbf{i}^l(u)\}_{l=1}^N$ with Eq. 6
        Update $\phi$ via adversarial training
    else
        Update $\psi$ and $\theta$ via adversarial training
    end if
    $iter \leftarrow iter + 1$
end while
return $\phi, \psi, \theta$
```

E EXPERIMENTAL DETAILS

In this section, we describe the detailed experimental environments of SPI-GAN. We build our experiments on top of (Kang et al., 2022)

E.1 EXPERIMENTAL ENVIRONMENTS

Our software and hardware environments are as follows: UBUNTU 18.04 LTS, PYTHON 3.9.7, PYTORCH 1.10.0, CUDA 11.1, NVIDIA Driver 417.22, i9 CPU, NVIDIA RTX A5000, and NVIDIA RTX A6000.

E.2 TARGET DIFFUSION MODEL

Our model uses a forward SDE to transform an image ($\mathbf{x}_0$) into a noise vector ($\mathbf{x}_T$). When generating a noise vector, we use the forward equation of VP-SDE for its high efficacy/effectiveness. The $\beta(t)$ function of VP-SDE is as follows:

$$\beta(t) = \beta_{\text{min}} + t(\beta_{\text{max}} - \beta_{\text{min}}), \tag{15}$$

where β_{max} = 20, β_{min} = 0.1, and $t' := \frac{t}{T}$ which is normalized from $t \in \{0, 1, \ldots, T\}$ to $[0, 1]$. Under these conditions, (Song et al., 2021c, Appendix B) proves that the noise vector at $t' = 1$ ($\mathbf{x}_T$) follows a unit Gaussian distribution.

E.3 DATA AUGMENTATION

Our model uses the adaptive discriminator augmentation (ADA) (Karras et al., 2020a), which has shown good performance in StyleGAN2.[2] The ADA applies image augmentation adaptively to training the discriminator. We can determine the maximum degree of the data augmentation, which is known as an ADA target, and the number of the ADA learning can be determined through the ADA interval. We also apply mixing regularization (λ_{mixing}) to encourage the styles to localize. Mixing regularization determines how many percent of the generated images are generated from two noisy images during training (a.k.a, style mixing). There are hyperparameters for the data augmentation in Table 8.

E.4 HYPERPARAMETERS

We list all the key hyperparameters in our experiments for each dataset. Our supplementary material accompanies some trained checkpoints and one can easily reproduce.

Table 8: Hyperparameters set for SPI-GAN.

		CIFAR-10	CelebA-HQ-256	LSUN-Church-256
Augmentation	ADA target	0.6	0.6	0.6
	ADA interval	4	4	4
	λ_{mixing} (%)	0	90	90
Architecture	Mapping network	1	7	7
	Discriminator	Original	Residual	Residual
Optimizer	Learning rate for generator	0.0025	0.0025	0.0025
	Learning rate for discriminator	0.0025	0.0025	0.0025
	EMA	0.999	0.999	0.999
	ODE Solver	4th order Runge–Kutta		
Regularization	Lazy generator	4	4	4
	Lazy discriminator	16	16	16
	λ_{R_1}	0.01	10	10
	λ_{path}	0	2	2

E.5 TRAINING TIME

The training time for each 1024 CIFAR-10 images is around 32.0s for SPI-GAN and around 45.6s for Diffusion-GAN using four NVIDIA A5000 GPUs.

E.6 ADDITIONAL ABLATION STUDIES

We report additional ablation studies to show the superiority of our model. First, the result of SPI-GAN without a mapping network is reported in Table 9. To generate a noise image $\hat{\mathbf{i}}(u)$ without

---

[2] `https://github.com/NVlabs/stylegan2` (Nvidia Source Code License)

a mapping network, u is given as a conditional input to the generator. Second, give time point u as a conditional input to the SPI-GAN generator. The result is in Table 10. SPI-GAN shows better performance than both ablation studies.

Table 9: Ablation study for mapping network

MODEL	FID
SPI-GAN (W/O MAPPING NETWORK)	5.72
SPI-GAN	3.01

Table 10: Ablation study for condition u

LAYER	DESIGN
SPI-GAN (CONDITION u TO GENERATOR)	3.03
SPI-GAN	3.01

F VISUALIZATION

We introduce interpolation and several high-resolution generated samples.

F.1 INTERPOLATION

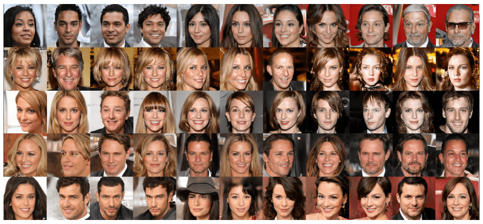

Figure 11: Generation by interpolating $\mathbf{h}(1) = (1-a)\mathbf{h}(1)' + a\mathbf{h}(1)''$, where $0 \leq a \leq 1$.

F.2 CIFAR-10

Figure 12: Qualitative results on CIFAR-10.

F.3 QUALITATIVE RESULTS ON HIGH RESOLUTION DATASETS

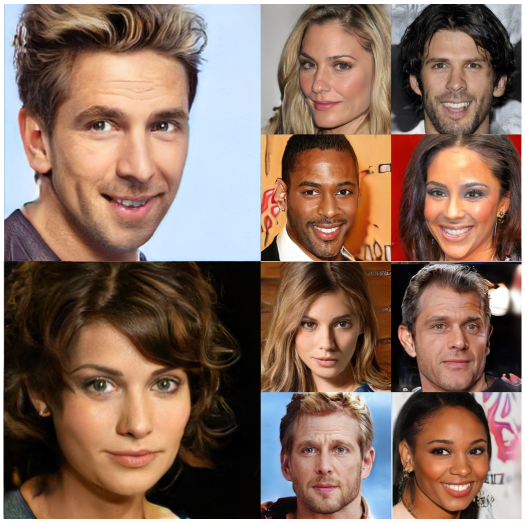

Figure 13: CelebA-HQ-256

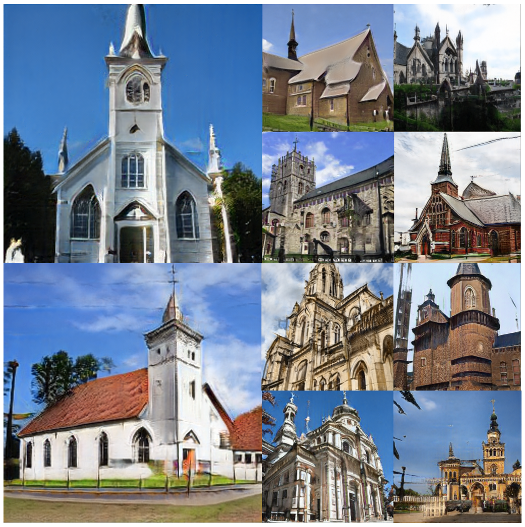

Figure 14: LSUN-Church-256

F.4 COMPARISON OF GENERATED DIFFUSION PROCESSES AND REAL DIFFUSION PROCESSES

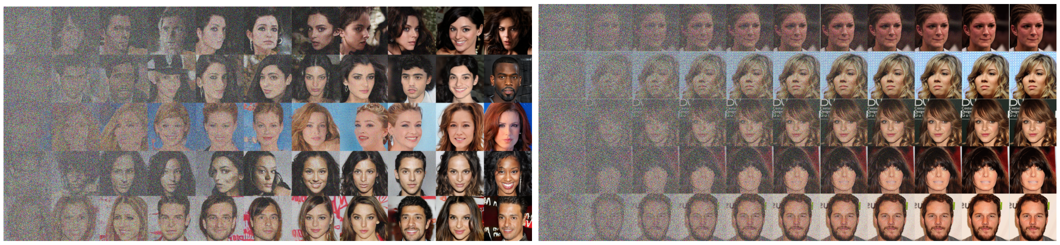

Figure 15: Comparing $\hat{\mathbf{i}}(u)$ and $\mathbf{i}(u)$. **Left:** Given a fixed $\mathbf{z}$, $\hat{\mathbf{i}}(u)$ is generated from SPI-GAN by varying the latent vector from $\mathbf{h}(0)$ to $\mathbf{h}(1)$. **Right:** Diffusion process of original images