A MATHEMATICAL DETAILS OF DIFFUSION MODELS

A.1 STOCHASTIC DPMS

In Eq. (1), we use $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t)$ and $\boldsymbol{\sigma}_t$ as a high-level abstraction to represent each reverse step t (T is the total number of steps) of stochastic DPMs. In Eq. (1), we define the sampling of $\boldsymbol{x}$ as

$$\begin{aligned}
&\boldsymbol{z} := \left(\boldsymbol{x}_T \oplus \boldsymbol{\epsilon}_T \oplus \cdots \oplus \boldsymbol{\epsilon}_2 \oplus \boldsymbol{\epsilon}_1\right) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \\
&\boldsymbol{x}_{T-1} = \boldsymbol{\mu}_T(\boldsymbol{x}_T, T) + \boldsymbol{\sigma}_T \odot \boldsymbol{\epsilon}_T, \\
&\boldsymbol{x}_{t-1} = \boldsymbol{\mu}_T(\boldsymbol{x}_t, t) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t, \quad T > t > 0, \\
&\boldsymbol{x} := \boldsymbol{x}_0.
\end{aligned} \tag{12}$$

To be self-contained, here we provide details of $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t)$ and $\boldsymbol{\sigma}_t$ for DDPM (Ho et al., 2020) and DDIM (Song et al., 2021a). Since the notations are not consistent in the two papers, we follow the notation in each paper respectively and use different colors to distinguish different notations. Also note that $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ stands for the neural network learned by DDPM and its variants, which should be distinguished from $\boldsymbol{\epsilon}_t$ used throughout this paper.

DDPM's $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t)$ and $\boldsymbol{\sigma}_t$: We follow the notation in Ho et al. (2020).

$$\boldsymbol{\mu}_T(\boldsymbol{x}_t, t) := \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right), \tag{13}$$

$$\boldsymbol{\sigma}_t := \begin{cases} \sqrt{\beta_t}\boldsymbol{I}, & \text{(option 1)} \\ \sqrt{\dfrac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}}\boldsymbol{I}, & \text{(option 2)} \\ \exp\left(\dfrac{\boldsymbol{v}_\theta(\boldsymbol{x}_t, t)}{2}\log\beta_t + \dfrac{\boldsymbol{I} - \boldsymbol{v}_\theta(\boldsymbol{x}_t, t)}{2}\log\dfrac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}\right). & \text{(option 3)} \end{cases} \tag{14}$$

DDIM's $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t)$ and $\boldsymbol{\sigma}_t$: We follow the notation in Song et al. (2021a).

$$\boldsymbol{\mu}_T(\boldsymbol{x}_t, t) := \sqrt{\alpha_{t-1}}\left(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t), \tag{15}$$

$$\boldsymbol{\sigma}_t := \sigma_t\boldsymbol{I}, \qquad \text{where } \sigma_t = \eta\sqrt{(1-\alpha_{t-1})/(1-\alpha_t)}\sqrt{1-\alpha_t/\alpha_{t-1}}, \tag{16}$$

where η is a hyper-parameter.

A.2 DETERMINISTIC DDIM

Deterministic DDIM's $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t)$: Deterministic DDIM is a special case of DDIM when $\eta = 0$. For details of other deterministic DPMs, please check the original papers.

A.3 SCORE-BASED GENERATIVE MODELING WITH SDE

Song et al. (2021b) proposed a unified view of DDPM and score matching with Langevin dynamics (SMLD) as different stochastic differential equations (SDEs). Since the randomness in their sampling algorithms purely come from Gaussian noise, we can incorporate their models and sampling methods into our framework. As a demonstration, we show how to define $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t)$ and $\boldsymbol{\sigma}_t$ for their predictor-only sampling with reverse diffusion samplers. Given a forward SDE:

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}, t)\mathrm{d}t + \boldsymbol{\sigma}(t)\odot\mathrm{d}\boldsymbol{w}, \tag{17}$$

the reverse-time SDE is

$$\mathrm{d}\boldsymbol{x} = [\boldsymbol{f}(\boldsymbol{x}, t) - \boldsymbol{\sigma}(t)^2\odot\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x})]\mathrm{d}t + \boldsymbol{\sigma}(t)\odot\mathrm{d}\bar{\boldsymbol{w}}. \tag{18}$$

Suppose the forward SDE is discretized in the following form:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{f}_t(\boldsymbol{x}_t) + \boldsymbol{\sigma}_t\odot\boldsymbol{z}_t, \quad t = 0, \ldots, T-1, \quad \boldsymbol{z}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}). \tag{19}$$

Reverse diffusion samplers discretize the reserve-time SDE in a similar form:

$$\boldsymbol{x}_{t-1} = \boldsymbol{x}_t - \boldsymbol{f}_t(\boldsymbol{x}_t) + \boldsymbol{\sigma}_t^2\odot\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) + \boldsymbol{\sigma}_t\odot\boldsymbol{\epsilon}_t, \quad t = 1, \ldots, T, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \tag{20}$$

where $\boldsymbol{s}_{\boldsymbol{\theta}}$ is a neural network trained to match the score $\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x})$. By comparing Eq. (12) and Eq. (20), we have $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t) := \boldsymbol{x}_t - \boldsymbol{f}_t(\boldsymbol{x}_t) + \boldsymbol{\sigma}_t^2\odot\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$.

A.4 DDGAN

In Eq. (5), we use $\boldsymbol{\mu}_T(\boldsymbol{x}_t, \boldsymbol{z}_t, t)$ and $\boldsymbol{\sigma}_t$ as high-level abstractions to represent each reverse step t (T is the total number of steps) of DDGAN. The generation process is defined as

$$\begin{aligned}
&\boldsymbol{z} := \left(\boldsymbol{x}_T \oplus \boldsymbol{z}_T \oplus \boldsymbol{\epsilon}_T \oplus \cdots \oplus \boldsymbol{z}_2 \oplus \boldsymbol{\epsilon}_2 \oplus \boldsymbol{z}_1\right) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \\
&\boldsymbol{x}_{T-1} = \boldsymbol{\mu}_T(\boldsymbol{x}_T, \boldsymbol{z}_T, T) + \boldsymbol{\sigma}_T \odot \boldsymbol{\epsilon}_T, \\
&\boldsymbol{x}_{t-1} = \boldsymbol{\mu}_T(\boldsymbol{x}_t, \boldsymbol{z}_t, t) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t, \quad T > t > 1, \\
&\boldsymbol{x} := \boldsymbol{x}_0 = \boldsymbol{\mu}_T(\boldsymbol{x}_1, \boldsymbol{z}_1, 1).
\end{aligned} \tag{21}$$

To be self-contained, here we provide details of $\boldsymbol{\mu}_T(\boldsymbol{x}_t, \boldsymbol{z}_t, t)$ and $\boldsymbol{\sigma}_t$ of DDGAN.

DDGAN's $\boldsymbol{\mu}_T(\boldsymbol{x}_t, \boldsymbol{z}_t, t)$ and $\boldsymbol{\sigma}_t$: We follow the notation in Xiao et al. (2022) and Ho et al. (2020).

$$\boldsymbol{\mu}_T(\boldsymbol{x}_t, \boldsymbol{z}_t, t) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} G_\theta(\boldsymbol{x}_t, \boldsymbol{z}_t, t) + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t, \tag{22}$$

$$\boldsymbol{\sigma}_t := \sqrt{\frac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}}\boldsymbol{I}, \tag{23}$$

where $G_\theta(\boldsymbol{x}_t, \boldsymbol{z}_t, t)$ is a conditional GAN learned by DDGAN, which should be distinguished from the deterministic mapping G used throughout this paper.

Algorithm 2: DPM-Encoder

Input: an image $\boldsymbol{x} := \boldsymbol{x}_0$, a pre-trained stochastic DPM with $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t)$, $\boldsymbol{\sigma}_t$, and $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$
1. Sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T-1}, \boldsymbol{x}_T \sim q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$
2. $\boldsymbol{z} = \boldsymbol{x}_T$
for $t = T, \ldots, 1$ **do**
  3. $\boldsymbol{\epsilon}_t = \left(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_t, t)\right)/\boldsymbol{\sigma}_t$
  4. $\boldsymbol{z} = \boldsymbol{z} \oplus \boldsymbol{\epsilon}_t$
5. **Output:** $\boldsymbol{z}$

B Mathematical Details of DPM-Encoder

In this section, we provide details of our DPM-Encoder introduced in Section 3.2, which samples $\boldsymbol{z} \sim \mathrm{DPMEnc}(\boldsymbol{z}|\boldsymbol{x}, G)$. For each image $\boldsymbol{x} := \boldsymbol{x}_0$, stochastic DPMs define a posterior distribution over the noisy images $\boldsymbol{x}_{1:T}$, denoted as $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ (Ho et al., 2020; Song et al., 2021a). To be self-contained, we provide details of this posterior distribution for different diffusion models.

DDPM's posterior $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$: We follow the notation in Ho et al. (2020).

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) := \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}), \quad q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) := \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \mathbf{I}\right). \tag{24}$$

DDIM's posterior $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$: We follow the notation in Song et al. (2021a).

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) := q(\boldsymbol{x}_T|\boldsymbol{x}_0)\prod_{t=2}^{T} q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0), \tag{25}$$

$$q(\boldsymbol{x}_T|\boldsymbol{x}_0) = \mathcal{N}(\sqrt{\alpha_T}\boldsymbol{x}_0, (1-\alpha_T)\,\boldsymbol{I}), \tag{26}$$

$$\begin{aligned}
q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) &= \mathcal{N}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}}, \sigma_t^2\boldsymbol{I}\right), \\
&\text{where } \sigma_t = \eta\sqrt{(1-\alpha_{t-1})\,/\,(1-\alpha_t)}\sqrt{1-\alpha_t/\alpha_{t-1}}.
\end{aligned} \tag{27}$$

Based on the posterior distribution $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$, DPM-Encoder samples the latent code $\boldsymbol{z}$ by first sampling noisy images $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ from $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ and computing the $\boldsymbol{\epsilon}_t$ according to Eq. (1) and

Eq. (12). Formally, we define the sampling process $\boldsymbol{z} \sim \mathrm{DPMEnc}(\boldsymbol{z}|\boldsymbol{x}, G)$ as

$$\begin{aligned} \boldsymbol{x}_1, \dots, \boldsymbol{x}_{T-1}, \boldsymbol{x}_T \sim q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0), \quad \boldsymbol{\epsilon}_t = \big(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_t, t)\big)/\boldsymbol{\sigma}_t, \quad t = T, \dots, 1, \\ \boldsymbol{z} := \big(\boldsymbol{x}_T \oplus \boldsymbol{\epsilon}_T \oplus \cdots \oplus \boldsymbol{\epsilon}_2 \oplus \boldsymbol{\epsilon}_1\big). \end{aligned} \tag{28}$$

DPM-Encoder guarantees perfect reconstruction. The proof is straightforward, provided as follows.

Proposition 1. *(Invertibility of DPM-Encoder) For each $\boldsymbol{z} \sim \mathrm{DPMEnc}(\boldsymbol{z}|\boldsymbol{x}, G)$ defined in Eq. (28), we have $\boldsymbol{x} = \bar{\boldsymbol{x}} := G(\boldsymbol{z})$, where $\bar{\boldsymbol{x}} := G(\boldsymbol{z})$ is defined as*

$$\begin{aligned} \bar{\boldsymbol{x}}_{T-1} &= \boldsymbol{\mu}_T(\boldsymbol{x}_T, T) + \boldsymbol{\sigma}_T \odot \boldsymbol{\epsilon}_T, \\ \bar{\boldsymbol{x}}_{t-1} &= \boldsymbol{\mu}_T(\bar{\boldsymbol{x}}_t, t) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t, \quad T > t > 0, \\ \bar{\boldsymbol{x}} &:= \bar{\boldsymbol{x}}_0. \end{aligned} \tag{29}$$

Proof. We prove $\bar{\boldsymbol{x}}_t = \boldsymbol{x}_t$ for all $T - 1 \geq t \geq 0$ by induction. The proposition holds when $\bar{\boldsymbol{x}}_0 = \boldsymbol{x}_0$. To begin with, $\bar{\boldsymbol{x}}_{T-1} = \boldsymbol{x}_{T-1}$ because

$$\bar{\boldsymbol{x}}_{T-1} = \boldsymbol{\mu}_T(\boldsymbol{x}_T, T) + \boldsymbol{\sigma}_T \odot \boldsymbol{\epsilon}_T \tag{30}$$

$$= \boldsymbol{\mu}_T(\boldsymbol{x}_T, T) + \boldsymbol{\sigma}_T \odot \big(\boldsymbol{x}_{T-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_T, T)\big)/\boldsymbol{\sigma}_T = \boldsymbol{x}_{T-1}. \tag{31}$$

For $T - 1 \geq t > 0$, when $\bar{\boldsymbol{x}}_t = \boldsymbol{x}_t$, we have

$$\bar{\boldsymbol{x}}_{t-1} = \boldsymbol{\mu}_T(\bar{\boldsymbol{x}}_t, t) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t \tag{32}$$

$$= \boldsymbol{\mu}_T(\boldsymbol{x}_t, t) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t \tag{33}$$

$$= \boldsymbol{\mu}_T(\boldsymbol{x}_t, t) + \boldsymbol{\sigma}_t \odot \big(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_t, t)\big)/\boldsymbol{\sigma}_t = \boldsymbol{x}_{t-1}. \tag{34}$$

□

C Experimental Details of Zero-Shot Image-to-Image Translation

Sources of images in the 150 tuples: For the zero-shot image-to-image translation experiment, we created a set of 150 tuples as task input, which include but are not limited to: (1) image generated by DALL·E 2 (Ramesh et al., 2022), (2) real images from Ruiz et al. (2022), (3) real images from (Hertz et al., 2022), (4) real images collected by the authors.

Per sample selection criterion: For each test sample, we allow each method to enumerate some combinations of hyperparameters (detailed below). To select the best combination for each sample, we used the directional CLIP score $\mathcal{S}_{\text{D-CLIP}}$ as the criterion (higher is better).

DDIB: DDIB edits images by using a deterministic DPM conditioned on the source text $\boldsymbol{t}$ to encode the source image, followed by decoding conditioned on the target text $\hat{\boldsymbol{t}}$. We used the deterministic DDIM sampler with 100 steps. We set the classifier-free guidance of the encoding step as 1; we enumerated the classifier-free guidance of the decoding step as $\{1, 1.5, 2, 3, 4, 5\}$.

SDEdit: SDEdit edits images by adding noise to the original image (the encoding step), followed by denoising the noised image with a diffusion model trained on the target domain (the decoding step). For zero-shot image-to-image translation, the decoding step of SDEdit uses the text-to-image diffusion model conditioned on the target image $\hat{\boldsymbol{t}}$. Notably, SDEdit does not provide a way to take the source text $\boldsymbol{t}$ as input. We used the DDIM sampler ($\eta = 0.1$) with 100 steps. We enumerated the classifier-free guidance of the decoding step as $\{1, 1.5, 2, 3, 4, 5\}$; we enumerated the encoding step as $\{15, 20, 25, 30, 40, 50\}$; we ran 15 trials for each hyperparameter combination.

CycleDiffusion: For our CycleDiffusion, we used the DDIM sampler ($\eta = 0.1$) with 100 steps. We set the classifier-free guidance of the encoding process as 1; we enumerated the classifier-free guidance of the decoding step as $\{1, 1.5, 2, 3, 4, 5\}$; we enumerated the early stopping step T_{es} as $\{15, 20, 25, 30, 40, 50\}$; we ran 15 trials for each hyperparameter combination.

D Resources

Our experiments used publicly available pre-trained checkpoints (except for the diffusion models trained by us on AFHQ Cat and Wild; see Section 4). Each experiment was run on one NVIDIA RTX A4000 (16G) / RTX A6000 (48G) / A100 (40G) GPU. We will make our code, configuration files, and experiment commands publicly available.

E ADDITIONAL RESULTS FOR ZERO-SHOT IMAGE-TO-IMAGE TRANSLATION

Figure 7 provides a qualitative comparison for zero-shot image-to-image translation. Compared with DDIB and SDEdit, CycleDiffusion greatly improves the faithfulness to the source image.

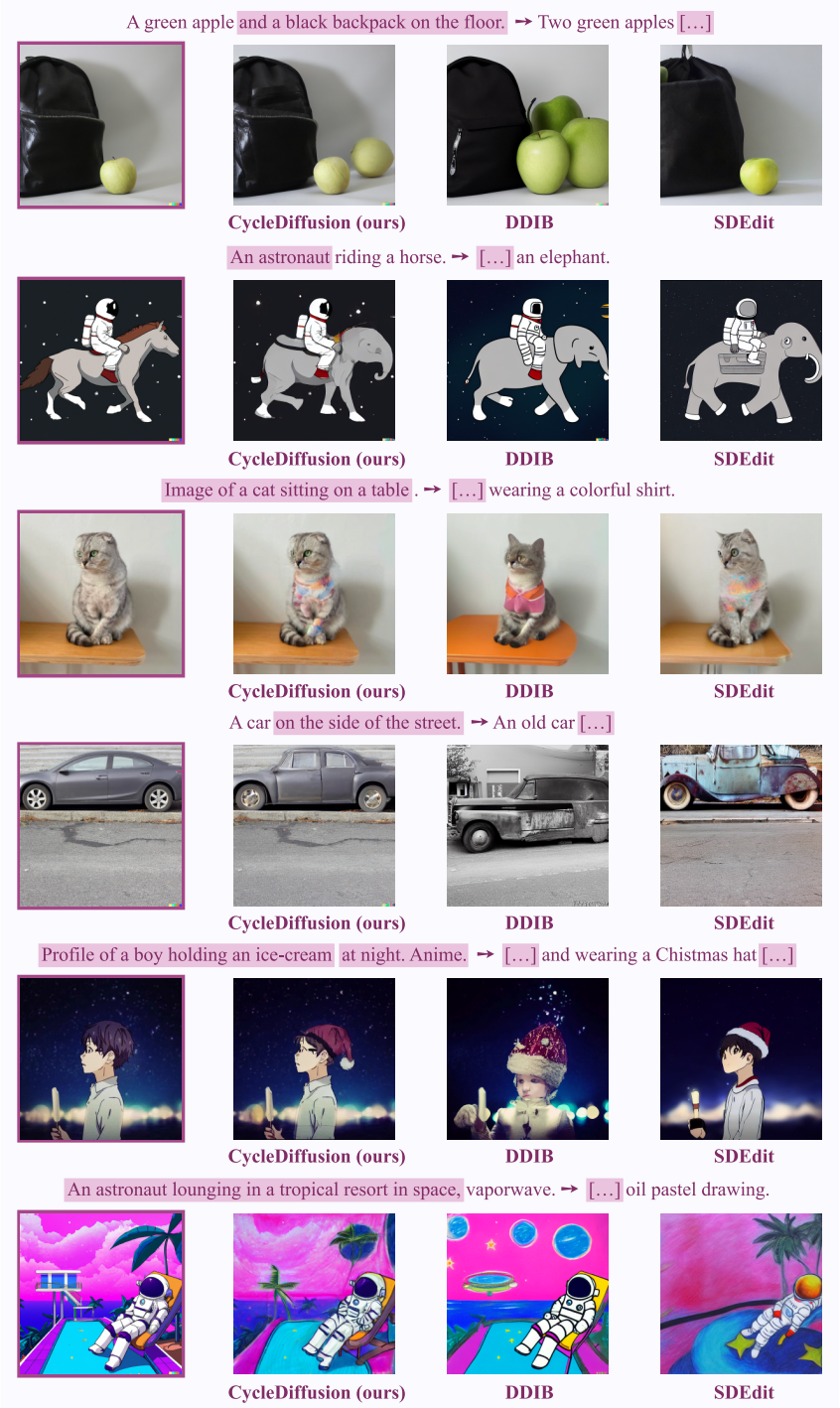

Figure 7: Samples for zero-shot image-to-image translation. Notations follow Figure 3. Compared with DDIB and SDEdit, CycleDiffusion greatly improves the faithfulness to the source image.

F LOCAL EDITING DDIM'S HIGH-DIMENSIONAL LATENT CODE

Local editing of low-dimensional latent code has been shown to be useful for semantic-level image manipulation (Shen et al., 2022). However, it is unclear whether we can perform semantic-level image manipulation via local editing in the high-dimensional latent space diffusion models. Note that this is different from mask-then-inpaint (Ramesh et al., 2022), edit-with-scribbles (Meng et al., 2022), or domain adaptation (Kim et al., 2022b)). Notably, it does not need the classifier to be adapted to noisy images as done by the classifier guidance (Dhariwal & Nichol, 2021; Liu et al., 2021).

Given an image $\boldsymbol{x}_{ori}$, we encode it as $\boldsymbol{z}_{ori}$, edit it as $\boldsymbol{z}_{edit} = \boldsymbol{z}_{ori} + \boldsymbol{n}$, and compute the edited image $\boldsymbol{x}_{edit} = G(\boldsymbol{z}_{edit})$. To learn the vector $\boldsymbol{n}$ for a target class a, we optimize

$$\arg\min_{\|\boldsymbol{n}\|_2=r} \mathbb{E}_{\boldsymbol{z}_{ori}\sim p_{\boldsymbol{z}}(\boldsymbol{z}_{ori}),\ \boldsymbol{z}_{edit}=\boldsymbol{z}_{ori}+\boldsymbol{n}} \Big[-\lambda_{cls} \log P\big(a|G(\boldsymbol{z}_{edit})\big) - \cos\big\langle R(G(\boldsymbol{z}_{edit})), R(G(\boldsymbol{z}_{ori}))\big\rangle \Big], \quad (35)$$

where $P(\cdot|\boldsymbol{x})$ is a classifier trained on CelebA (Liu et al., 2015), and R is the IR-SE50 face embedding model (Deng et al., 2019) to preserve the identity. Empirically, we find that LDM-DDIM ($\eta = 0$) works the best for local editing, as shown in Figure 8.

Figure 8: Image manipulation by local editing of diffusion models' latent code. The diffusion model used here is the deterministic LDM-DDIM ($\eta = 0$).

Table 5: State-of-the-art generative models used in this paper. Notations: *struc.*, –, and $\oplus$ stand for *structure*, *no "latent codes"*,[2] and *progressive generation*, respectively.

	Model name	Latent prior	Objective	Architecture	Latent struc.	Resolution
Diffusion	DDPM (Ho et al., 2020) *etc.*	–	ELBO	{CNN, ViT}	–	256
	DDIM ($\eta = 0$) (Song et al., 2021a)	Gaussian	ELBO		spatial	256
	SN-DDPM (Bao et al., 2022)	–	ELBO		–	64
	ScoreSDE (Song et al., 2021b)	–	ELBO / SM		–	256 / 1024
	LDM (Rombach et al., 2022) *etc.*	diffusion	ELBO		spatial	256
	DiffAE (Preechakul et al., 2022)	diffusion	ELBO		hybrid	256
	DDGAN (Xiao et al., 2022)	–	hybrid		–	256
2D GAN	StyleGAN2 (Karras et al., 2020)	Gaussian	GAN	CNN	vector	1024
	StyleGAN-XL (Sauer et al., 2022)		GAN	CNN		256 – 1024
	StyleSwin (Zhang et al., 2022a)		GAN	ViT		256 / 1024
	BigGAN (Brock et al., 2019)		GAN	CNN		256
	Diffusion-GAN (Wang et al., 2022)		hybrid	CNN		1024
3D GAN	StyleNeRF (Gu et al., 2022)	Gaussian	GAN	NeRF $\oplus$ CNN	vector	256 – 1024
	GIRAFFE-HD (Xue et al., 2022)			NeRF $\oplus$ CNN		1024
	StyleSDF (Or-El et al., 2022)			SDF $\oplus$ CNN		512 / 1024
	EG3D (Chan et al., 2022)			TriPl $\oplus$ CNN		512
VAE	NVAE (Vahdat & Kautz, 2020)	Gaussian	ELBO	CNN	spatial	256

G ADDITIONAL RESULTS FOR PLUG-AND-PLAY GUIDANCE

Seen in Table 5 is a summary of generative models unified as deterministic mappings in this paper. Different models have different training objectives, model architectures, and structures of "latent code"[2]. Most of the listed models are included in our experiments. Table 6 and Table 7 provide a more detailed version of the results (for some generative models) seen in Figure 5. Specifically, we investigated different configurations of various diffusion models and GANs. In Figure 9, we provide several image samples for ID-controlled sampling from pre-trained generative models. Consistent with Table 4, diffusion models have better coverage of individuals than 2D/3D GANs.

Table 6: CLIP experiments of models that have different configurations. Numbers under each model stand for the image resolution; *trunc.* $\phi = 0.7$ stands for the truncation trick (Karras et al., 2019) with truncation coefficient $\phi = 0.7$. The reported metric is the CLIP score (larger is better), the same as Figure 5. $^\heartsuit$ and $^\spadesuit$ stand for the configuration plotted in Figure 5.

Text t (Figure 5)	$\{$ 👶 $\cdots\}^\heartsuit$					$\{$ 👩🏿 👱 $\cdots\}^\spadesuit$				
Control strength λ_{CLIP}	100	300	500	700	1000	100	300	500	700	1000
LDM-DDIM ($\eta = 0$)										
256 ($T_g = 10$)$^{\heartsuit\spadesuit}$	0.258	0.276	0.283	0.288	0.290	0.269	0.283	0.296	0.300	0.308
256 ($T_g = 5$)	0.257	0.280	0.283	0.285	0.287	0.269	0.283	0.292	0.298	0.304
DiffAE										
256 ($T_g = 10$)$^{\heartsuit\spadesuit}$	0.266	0.287	0.291	0.294	0.294	0.270	0.296	0.307	0.314	0.319
128 ($T_g = 3$)	0.259	0.284	0.289	0.290	0.292	0.256	0.271	0.286	0.293	0.298
128 ($T_g = 3$, z_T only)	0.256	0.289	0.289	0.290	0.295	0.256	0.270	0.285	0.293	0.297
StyleGAN2										
1024$^{\heartsuit\spadesuit}$	0.273	0.293	0.296	0.296	0.298	0.275	0.302	0.308	0.311	0.312
1024 (*trunc.* $\phi = 0.7$)	0.267	0.287	0.291	0.293	0.293	0.267	0.291	0.299	0.301	0.303
StyleGAN-XL										
1024$^{\heartsuit\spadesuit}$	0.270	0.291	0.294	0.295	0.295	0.273	0.299	0.308	0.312	0.313
1024 (*trunc.* $\phi = 0.7$)	0.263	0.283	0.287	0.289	0.290	0.265	0.284	0.292	0.295	0.297
512 (*trunc.* $\phi = 0.7$)	0.263	0.282	0.286	0.288	0.289	0.263	0.284	0.293	0.296	0.300
256 (*trunc.* $\phi = 0.7$)	0.262	0.281	0.284	0.287	0.289	0.259	0.281	0.291	0.295	0.299
StyleSwin										
1024$^{\heartsuit\spadesuit}$	0.266	0.279	0.278	0.276	0.268	0.273	0.291	0.296	0.295	0.294
256	0.262	0.282	0.283	0.283	0.281	0.267	0.285	0.290	0.293	0.293
1024 (*trunc.* $\phi = 0.7$)	0.265	0.284	0.287	0.288	0.288	0.264	0.279	0.292	0.297	0.300
256 (*trunc.* $\phi = 0.7$)	0.259	0.278	0.281	0.275	0.273	0.261	0.276	0.281	0.284	0.281
Diffusion-GAN										
1024$^{\heartsuit\spadesuit}$	0.278	0.295	0.298	0.298	0.299	0.270	0.297	0.305	0.307	0.308
1024 (*trunc.* $\phi = 0.7$)	0.273	0.294	0.294	0.300	0.301	0.262	0.286	0.300	0.305	0.309
StyleNeRF										
1024	0.246	0.271	0.283	0.288	0.291	0.264	0.291	0.303	0.307	0.311
256	0.234	0.240	0.247	0.252	0.259	0.260	0.275	0.291	0.298	0.303
1024 (*trunc.* $\phi = 0.7$)	0.243	0.266	0.277	0.283	0.287	0.260	0.280	0.291	0.296	0.300
256 (*trunc.* $\phi = 0.7$)	0.229	0.235	0.239	0.243	0.249	0.255	0.267	0.282	0.290	0.295
StyleSDF										
1024$^{\heartsuit\spadesuit}$	0.275	0.288	0.286	0.282	–	0.270	0.290	0.292	0.291	–
1024 (*trunc.* $\phi = 0.7$)	0.267	0.283	0.284	0.279	–	0.261	0.270	0.273	0.273	–
EG3D										
512$^{\heartsuit\spadesuit}$	0.277	0.292	0.294	0.295	0.294	0.272	0.298	0.305	0.307	0.310
512 (*trunc.* $\phi = 0.7$)	0.277	0.270	0.276	0.280	0.283	0.265	0.285	0.293	0.296	0.298

Figure 9: Image samples for the face ID experiment in Table 4.

Table 7: CLIP experiments of models that have different configurations. Numbers under each model stand for the image resolution; *trunc.* $\phi = 0.7$ stands for the truncation trick (Karras et al., 2019) with truncation coefficient $\phi = 0.7$. The reported metric is the CLIP score (larger is better), the same as Figure 5. $^\heartsuit$ and $^\spadesuit$ stand for the configuration plotted in Figure 5.

Text t (Figure 5)	🕶 $^\heartsuit$					🕶 🎩 $^\spadesuit$				
Control strength λ_{CLIP}	100	300	500	700	1000	100	300	500	700	1000
LDM-DDIM ($\eta = 0$)										
256 ($T_g = 10$)$^{\heartsuit\spadesuit}$	0.275	0.290	0.297	0.301	0.307	0.252	0.288	0.312	0.326	0.343
256 ($T_g = 5$)	0.273	0.289	0.300	0.305	0.310	0.250	0.288	0.315	0.329	0.343
DiffAE										
256 ($T_g = 10$)$^{\heartsuit\spadesuit}$	0.275	0.290	0.297	0.303	0.307	0.250	0.287	0.308	0.320	0.328
128 ($T_g = 3$)	0.265	0.281	0.288	0.293	0.298	0.240	0.273	0.300	0.314	0.326
128 ($T_g = 3$, z_T only)	0.263	0.280	0.288	0.291	0.296	0.240	0.275	0.300	0.313	0.324
StyleGAN2										
1024$^{\heartsuit\spadesuit}$	0.279	0.294	0.300	0.303	0.304	0.264	0.299	0.313	0.319	0.321
1024 (*trunc.* $\phi = 0.7$)	0.278	0.291	0.297	0.300	0.303	0.255	0.286	0.303	0.311	0.316
StyleGAN-XL										
1024$^{\heartsuit\spadesuit}$	0.282	0.299	0.306	0.310	0.310	0.260	0.296	0.301	0.304	0.305
1024 (*trunc.* $\phi = 0.7$)	0.281	0.297	0.303	0.305	0.309	0.253	0.279	0.287	0.288	0.291
512 (*trunc.* $\phi = 0.7$)	0.280	0.296	0.301	0.305	0.307	0.250	0.282	0.296	0.300	0.303
256 (*trunc.* $\phi = 0.7$)	0.275	0.290	0.297	0.299	0.303	0.251	0.283	0.295	0.300	0.304
StyleSwin										
1024$^\heartsuit$	0.276	0.282	0.284	0.281	0.278	0.251	0.263	0.258	0.255	0.247
256$^\spadesuit$	0.273	0.281	0.285	0.284	0.281	0.256	0.277	0.281	0.277	0.275
1024 (*trunc.* $\phi = 0.7$)	0.276	0.284	0.286	0.290	0.288	0.243	0.263	0.274	0.277	0.275
256 (*trunc.* $\phi = 0.7$)	0.272	0.280	0.281	0.282	0.281	0.248	0.267	0.275	0.274	0.269
Diffusion-GAN										
1024$^{\heartsuit\spadesuit}$	0.278	0.294	0.301	0.303	0.306	0.262	0.288	0.298	0.301	0.302
1024 (*trunc.* $\phi = 0.7$)	0.277	0.291	0.300	0.291	0.308	0.249	0.279	0.289	0.296	0.301
StyleNeRF										
1024	0.268	0.277	0.282	0.285	0.287	0.238	0.252	0.262	0.272	0.281
256	0.264	0.272	0.277	0.280	0.283	0.235	0.244	0.252	0.256	0.263
1024 (*trunc.* $\phi = 0.7$)	0.268	0.276	0.281	0.284	0.286	0.233	0.244	0.253	0.261	0.270
256 (*trunc.* $\phi = 0.7$)	0.264	0.271	0.277	0.280	0.282	0.232	0.238	0.246	0.251	0.255
StyleSDF										
1024$^{\heartsuit\spadesuit}$	0.275	0.278	0.273	–	–	0.253	0.259	–	–	–
1024 (*trunc.* $\phi = 0.7$)	0.273	0.279	0.275	–	–	0.242	0.252	0.248	–	–
EG3D										
512$^{\heartsuit\spadesuit}$	0.284	0.297	0.303	0.305	0.308	0.257	0.284	0.287	0.287	0.281
512 (*trunc.* $\phi = 0.7$)	0.282	0.295	0.300	0.301	0.305	0.246	0.268	0.276	0.278	0.276

H Societal Impact

In general, improved generative modeling makes it easier to generate fake media (e.g., DeepFakes; Westerlund, 2019; Vaccari & Chadwick, 2020) and privacy leaks (e.g., identity-conditioned human face synthesis, information leaks from large-scale pre-training data of text-to-image diffusion models). Additionally, in the particular case of this paper, one could encounter biases image editing as a result of applying CycleDiffusion to text-to-image diffusion models that reflect the natural biases in large text-image pre-training data. On the other hand, improved generative modeling can bring benefits to synthesis of humans and new ways of human communication in AR/VR. Moreover, we point out that there exist many current research works and tools that can efficiently detect fake media or can manage privacy leaks during pre-training. We encourage researchers and practitioners to consider these risks and remedies when using the methods developed in this paper.