Making Text-to-Image Diffusion Models Zero-Shot Image-to-Image Editors by Inferring "Random Seeds"

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent text-to-image diffusion models trained on large-scale data achieve remarkable performance on text-conditioned image synthesis (e.g., GLIDE, DALL·E-2, Imagen, Stable Diffusion). This paper presents an embarrassingly simple method to use these text-to-image diffusion models as **zero-shot** image-to-image editors. Our method, **CycleDiffusion**, is based on a recent finding that, when the "random seed" is fixed, sampling from two diffusion model distributions will produce images with minimal differences, and the core of our idea is to infer the "random seed" that is likely to produce a source image conditioned on a source text. We formalize the "random seed" as a sequence of isometric Gaussian noises that we reformulate as diffusion models' latent code. Using the "random seed" inferred from the source text-image pair, we generate a target image conditioned a target text. Experiments show that CycleDiffusion can minimally edit the image in a zero-shot manner.

1 Introduction

We think the randomness in diffusion models is like magic. As demonstrated by recent works (e.g., Figure 11 in [12], copied as Figure 2), given a fixed random seed, a text-to-image diffusion model will generate images with minimal differences when conditioned on two text prompts. In this paper, we show how to leverage this magical observation to make text-to-image diffusion models **zero-shot** image-to-image editors, without any assumptions on the model architecture.

We first formalize "random seeds" by redefining diffusion models' latent space (Section 2.1). The widely-adopted characterization of diffusion models' latent code is a sequence of gradually denoised images [8, 22]. We provide an alternative formulation of diffusion models as a *deterministic* mapping from Gaussian latent codes $\boldsymbol{z}$ to images $\boldsymbol{x}$, where $\boldsymbol{z}$ is the concatenation of all isometric Gaussian noises in each reverse step. After this reformulation, diffusion models can be denoted as $\boldsymbol{x} = G(\boldsymbol{z})$. Since all randomness is in $\boldsymbol{z}$, fixing $\boldsymbol{z}$ is equivalent to fixing the "random seed".

We then study how to infer "random seeds" from a given image (Section 2.2), which is essentially an encoding problem: given a real image $\boldsymbol{x}$, can we sample $\boldsymbol{z}$ such that $\boldsymbol{x} = G(\boldsymbol{z})$? We provide an affirmative answer by proposing **DPM-Encoder**. DPM-Encoder is built upon the fact that one can always sample a sequence of noisy images from the predefined posterior of diffusion models. After sampling the noisy images, DPM-Encoder computes the Gaussian noises in each step by definition.

Finally, we propose **CycleDiffusion** (Section 2.3), a zero-shot image-to-image translation method based on DPM-Encoder. It first encodes the source image $\boldsymbol{x}$ as $\boldsymbol{z}$ with a text-to-image diffusion model conditioned on the source text $\boldsymbol{t}$; then it decodes $\boldsymbol{z}$ into the target image $\hat{\boldsymbol{x}}$ with the text-to-image diffusion model conditioned on the target text $\hat{\boldsymbol{t}}$.

Submitted to Score-based Method Workshop at 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

In our experiments, we re-purposed SDEdit [11] and DDIB [23] as our baseline. Results demonstrate that CycleDiffusion is better at minimally edit the image to satisfy the target text. Seen in Figure 1 are samples from our CycleDiffusion. We also provide results of CycleDiffusion applied to the traditional unpaired image-to-image translation task (e.g., from cats to dogs) in Appendix E, which show that CycleDiffusion outperforms previous methods based on GANs or diffusion models.

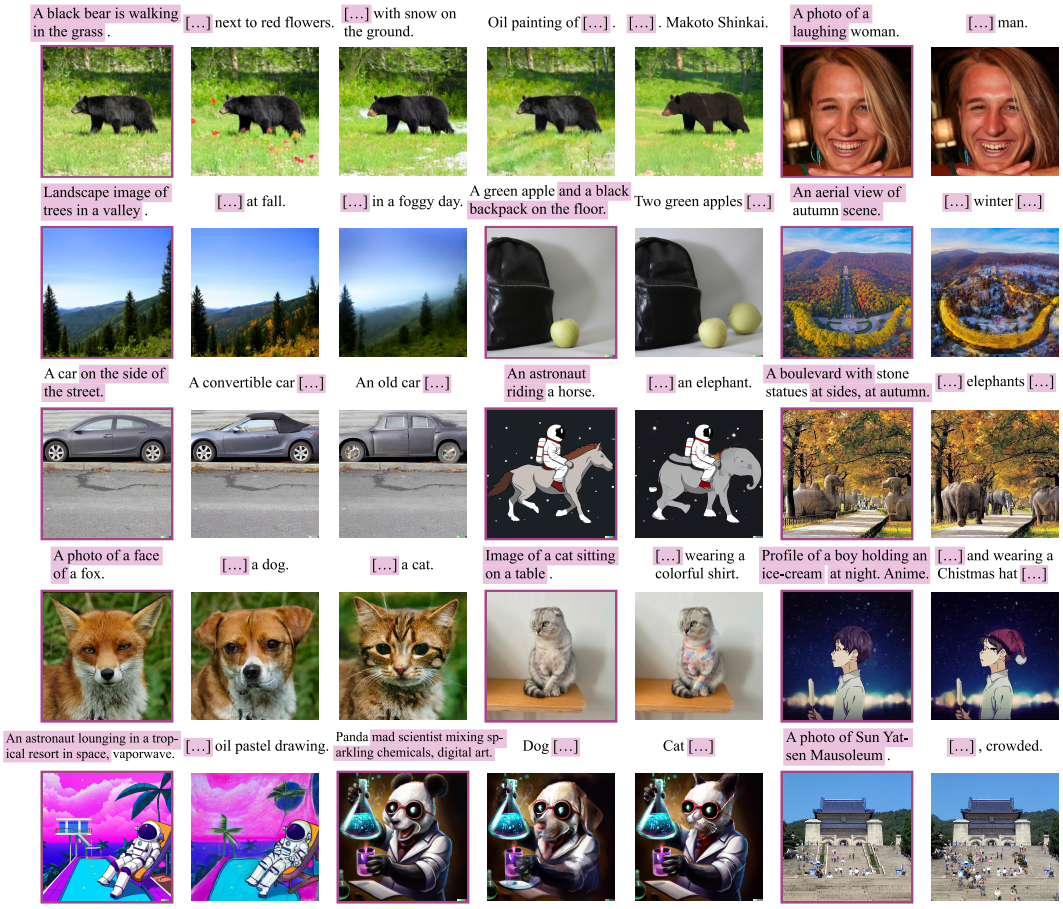

Figure 1: Text-to-image diffusion models can be zero-shot image-to-image editors. Source images $\boldsymbol{x}$ are displayed with a purple margin; the others are generated target images $\hat{\boldsymbol{x}}$. Overlapping text spans are marked in purple in source texts $\boldsymbol{t}$ and abbreviated as $[\ldots]$ in target texts $\hat{\boldsymbol{t}}$.

2 Method

2.1 Formalizing "Random Seeds" by Redefining Diffusion Models's Latent Space

In this section, we formalize "random seeds" as the redefined diffusion models' latent code. Diffusion models such as DDPMs [8], non-deterministic DDIMs [21], and score-based SDEs [22] generate images with a Markov chain structure, and we call them stochastic DPMs. These models have a mean estimator $\boldsymbol{\mu}_T$ (see Appendix A); given $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, the image $\boldsymbol{x} := \boldsymbol{x}_0$ is generated through $\boldsymbol{x}_{t-1} \sim \mathcal{N}(\boldsymbol{\mu}_T(\boldsymbol{x}_t, t), \mathrm{diag}(\boldsymbol{\sigma}_t^2))$. We can define the latent code $\boldsymbol{z}$ and the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ as

$$\boldsymbol{z} := \left(\boldsymbol{x}_T \oplus \boldsymbol{\epsilon}_T \oplus \cdots \oplus \boldsymbol{\epsilon}_1\right) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \quad \boldsymbol{x}_{t-1} = \boldsymbol{\mu}_T(\boldsymbol{x}_t, t) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t, \quad t = T, \ldots, 1, \tag{1}$$

where $\oplus$ is concatenation. Here, $\boldsymbol{z}$ has dimension $d = d_I \times (T+1)$, where d_I is the image dimension. Latent diffusion models (LDMs) [17] have becoming increasingly popular because of their efficiency. An LDM first uses a diffusion model G_{latent} to compute a "latent code" $\boldsymbol{z}_0 = G_{\text{latent}}(\boldsymbol{z})$,[1] which is then decoded as $\boldsymbol{x} = D(\boldsymbol{z}_0)$. Note that G_{latent} is an abstraction of the "latent-space" diffusion model.

[1]Quotation marks stand for "latent code" in the cited papers, different from our latent code $\boldsymbol{z}$ in Section 2.1.

2.2 DPM-Encoder: an Invertible Encoder for Diffusion Models

The previous section formalizes the "random seed" in stochastic DPMs as the latent code $\boldsymbol{z}$. However, to achieve zero-shot image-to-image translation, we need to infer $\boldsymbol{z}$ from a given real image $\boldsymbol{x}$. To this end, we propose **DPM-Encoder**, an invertible encoder for stochastic DPMs. For each image $\boldsymbol{x} := \boldsymbol{x}_0$, a stochastic DPM have a predefined posterior $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ [8, 21]. Based on $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ and Eq. (1), we can directly derive $\boldsymbol{z} \sim \mathrm{DPMEnc}(\boldsymbol{z}|\boldsymbol{x}, G)$ as (see details in Appendices A and B)

$$\begin{gathered}\boldsymbol{x}_1, \dots, \boldsymbol{x}_{T-1}, \boldsymbol{x}_T \sim q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0), \quad \boldsymbol{\epsilon}_t = \big(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_t, t)\big)/\boldsymbol{\sigma}_t, \quad t = T, \dots, 1, \\ \boldsymbol{z} := \big(\boldsymbol{x}_T \oplus \boldsymbol{\epsilon}_T \oplus \cdots \oplus \boldsymbol{\epsilon}_2 \oplus \boldsymbol{\epsilon}_1\big).\end{gathered} \tag{2}$$

DPM-Encoder also applies to latent diffusion models (LDMs). This is achieved by first using the LDM encoder E to encode $\boldsymbol{z}_0 = E(\boldsymbol{x})$ and then sampling $\boldsymbol{z} \sim \mathrm{DPMEnc}(\boldsymbol{z}|\boldsymbol{z}_0, G_{\text{latent}})$.

2.3 CycleDiffusion: Image-to-Image Translation with DPM-Encoder

Since DPM-Encoder can infer $\boldsymbol{z}$ from a given real image $\boldsymbol{x}$, we use it to build our **zero-shot** image-to-image translation method **CycleDiffusion**. Let $G_{\boldsymbol{t}}$ be a text-to-image diffusion model conditioned on text $\boldsymbol{t}$. The task input is a source image $\boldsymbol{x}$, a source text $\boldsymbol{t}$ describing the source image $\boldsymbol{x}$, and a target text $\hat{\boldsymbol{t}}$ describing the target image $\hat{\boldsymbol{x}}$ to be generated. Like the GAN-based UNIT method [10], CycleDiffusion encodes $\boldsymbol{x}$ as $\boldsymbol{z}$ with DPM-Encoder and decodes it as $\hat{\boldsymbol{x}} = G_{\hat{\boldsymbol{t}}}(\boldsymbol{z})$. Formally, we have

$$\boldsymbol{z} \sim \mathrm{DPMEnc}(\boldsymbol{z}|\boldsymbol{x}, G_{\boldsymbol{t}}), \quad \hat{\boldsymbol{x}} = G_{\hat{\boldsymbol{t}}}(\boldsymbol{z}). \tag{3}$$

Detailed algorithm descriptions can be found in Appendix D, where we also design different variants of CycleDiffusion, such as noise removal and early stopping, to further improve the performance. In Appendix C, we discuss how to apply CycleDiffusion to the traditional unpaired image-to-image translation, which aims at aligning images between two domains.

3 Experiments

This section provides experiments for **zero-shot** image-to-image translation. Following Section 2.3, we curated a set of 150 triplets $(\boldsymbol{x}, \boldsymbol{t}, \hat{\boldsymbol{t}})$ for this task, where $\boldsymbol{x}$ is the source image, $\boldsymbol{t}$ is the source text (e.g., "an aerial view of autumn scene." in Figure 1), and $\hat{\boldsymbol{t}}$ is the target text (e.g., "an aerial view of winter scene."). The target image to be generated is denoted as $\hat{\boldsymbol{x}}$. Notably, most experiments in [6] tried to edit images generated by Imagen [18] and assumed that the random seed is known, which do not follow our setting since we allow $\boldsymbol{x}$ to be a real image. Further, without open-sourced codes, we could not compare with their mask heuristics on editing real images. In light of these restrictions, we suggest a new protocol for zero-shot image-to-image editing.

Metrics: To evaluate the faithfulness to source images, we used PSNR and SSIM. To evaluate the authenticity to the target text, we used the CLIP score, i.e., the cosine similarity between normalized CLIP embeddings: $\mathcal{S}_{\text{CLIP}}(\hat{\boldsymbol{x}}|\hat{\boldsymbol{t}}) = \cos\big\langle \text{CLIP}_{\text{img}}(\hat{\boldsymbol{x}}), \text{CLIP}_{\text{text}}(\hat{\boldsymbol{t}})\big\rangle$. However, it should be noted that there is a trade-off between PSNR/SSIM and $\mathcal{S}_{\text{CLIP}}$: by copying the source image we get perfect PSNR/SSIM, and by ignoring the source image we get high $\mathcal{S}_{\text{CLIP}}$. Here, we propose to evaluate zero-shot image-to-image translation using directional CLIP score [14], i.e., the cosine similarity between the normalized differences of normalized CLIP embeddings caused by translation:

$$\mathcal{S}_{\text{D-CLIP}}(\hat{\boldsymbol{x}}|\boldsymbol{x}, \boldsymbol{t}, \hat{\boldsymbol{t}}) = \cos\Big\langle \text{CLIP}_{\text{img}}(\hat{\boldsymbol{x}}) - \text{CLIP}_{\text{img}}(\boldsymbol{x}), \text{CLIP}_{\text{text}}(\hat{\boldsymbol{t}}) - \text{CLIP}_{\text{text}}(\boldsymbol{t})\Big\rangle. \tag{4}$$

Baselines: The baselines include SDEdit [11] and DDIB [23]. We used the same hyperparameters for the baselines and CycleDiffusion whenever possible (see Appendix F).

Pre-trained text-to-image diffusion models: To investigate how the zero-shot performance changes with data size, data quality, and training details, we used the following models: (1) `LDM-400M`, a 1.45B-parameter model trained on LAION-400M [20], (2) `SD-v1-1`, a 0.98B-parameter Stable Diffusion model trained on LAION-5B [19], (3) `SD-v1-4`, finetuned from `SD-v1-1` for improved aesthetics and classifier-free guidance.

Seen in Table 1 are the results for zero-shot image-to-image translation. When the same pre-trained text-to-image model is used, CycleDiffusion achieves the best zero-shot performance. CycleDiffusion

excels at being faithful to the source image (i.e., PSNR and SSIM); by contrast, SDEdit and DDIB have comparable authenticity to the target text (i.e., $\mathcal{S}_{\text{CLIP}}$), but their outputs are far away from the source images (see Appendix G). For all methods, using `SD-v1-1` or `SD-v1-4` as the pre-trained text-to-image diffusion model results in better faithfulness than using `LDM-400M`. Figure 1 provides several samples from our CycleDiffusion, demonstrating that the edits are minimal and meaningful.

Table 1: Quantitative evaluation for **zero-shot** image-to-image translation.

	Method	$\mathcal{S}_{\text{CLIP}}\uparrow$	$\mathcal{S}_{\text{D-CLIP}}\uparrow$	PSNR$\uparrow$	SSIM$\uparrow$
`LDM-400M`	SDEdit [11]	0.332	0.264	13.68	0.390
	DDIB [23]	0.324	0.195	15.82	0.544
	CycleDiffusion w/ DDIM ($\eta = 0.1$; ours)	**0.333**	**0.275**	**18.72**	**0.625**
`SD-v1-1`	SDEdit [11]	**0.339**	0.248	15.50	0.498
	DDIB [23]	0.324	0.189	18.58	0.658
	CycleDiffusion w/ DDIM ($\eta = 0.1$; ours)	0.331	**0.262**	**21.98**	**0.731**
`SD-v1-4`	SDEdit [11]	**0.344**	0.258	15.93	0.512
	DDIB [23]	0.331	0.209	18.10	0.653
	CycleDiffusion w/ DDIM ($\eta = 0.1$; ours)	0.334	**0.272**	**21.92**	**0.731**

4 Related Work

Many recent methods for text-to-image synthesis are built upon diffusion models (e.g., GLIDE [12], DALL·E 2 [15], Imagen [18], Stable Diffusion [17]) or Vector Quantized (VQ) Transformers (e.g., DALL·E [16], CogView [4, 5], Parti [28]). This paper focuses on text-to-image diffusion models.

It has been observed that diffusion models tend to generate images with minimal differences when using the same random seed. This holds for both stochastic diffusion models [8, 21] and deterministic diffusion models [21, 22]. For deterministic diffusion models, [23] and [9] provided explanations based on optimal transport, while there is currently no interpretation for stochastic diffusion models to the best of our knowledge. In terms of the application, [23] proposed dual diffusion implicit bridge (DDIB) for unpaired image-to-image translation, built upon the finding for deterministic diffusion models, while our CycleDiffusion is an extension of it to general stochastic diffusion models.

A fundamental problem in generative modeling is to design an encoder that is efficient and invertible. GAN inversion learns an encoder for pre-trained GANs [27]. VAEs and normalizing flows have their encoders by design. [21] and [22] studied encoding for deterministic diffusion models. However, it remains unclear how to encode for stochastic diffusion models. Our DPM-Encoder fills this blank.

Besides random seeds, a concurrent work [6] found that fixing the cross attention map in Transformer-based text-to-image diffusion models further improves the minimality of changes. However, to edit *real* images, [6] further resorted to the deterministic DDIM or mask heuristics because "how to infer random seeds" for stochastic diffusion models was considered non-trivial. Since our DPM-Encoder-based CycleDiffusion infers the "random seed", it can be combined with the method in [6]. Since [6] and Imagen do not have open-sourced implementation, we leave this attempt for future work.

5 Conclusions

This paper provides a formal description of the recent discussion on the "random seed" in diffusion models. We formalize the "random seed" as a sequence of isometric Gaussian noises, defined as diffusion models' latent code. Based on this formalism, we then propose DPM-Encoder, which infers the "random seed", followed by our CycleDiffusion method, which is capable of zero-shot image-to-image translation with text-to-image diffusion models (and unpaired image-to-image translation with diffusion models independently trained on two domains; Appendix E). We hope our results will inspire future work on novel applications of text-to-image diffusion models. Moreover, it is worth noting that CycleDiffusion leverages the observation that a common latent space emerges from two similar image distributions, while the mechanism behind it still needs further exploration.

References

[1] Mikołaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. *ICLR*, 2018.

[2] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. ILVR: Conditioning method for denoising diffusion probabilistic models. *ICCV*, 2021.

[3] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN v2: Diverse image synthesis for multiple domains. *CVPR*, 2020.

[4] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, and Jie Tang. CogView: Mastering text-to-image generation via Transformers. *NeurIPS*, 2021.

[5] Ming Ding, Wendi Zheng, Wenyi Hong, and Jie Tang. CogView2: Faster and better text-to-image generation via hierarchical Transformers. *NeurIPS*, 2022.

[6] Amir Hertz, Ron Mokady, Jay M. Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *ArXiv*, 2022.

[7] 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. *NIPS*, 2017.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

[9] Valentin Khrulkov and I. Oseledets. Understanding DDPM latent codes through optimal transport. *ArXiv*, 2022.

[10] Ming-Yu Liu, Thomas M. Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NIPS*, 2017.

[11] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. *ICLR*, 2022.

[12] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. *ArXiv*, 2021.

[13] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. *ECCV*, 2020.

[14] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and D. Lischinski. StyleCLIP: Text-driven manipulation of StyleGAN imagery. *ICCV*, 2021.

[15] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *ArXiv*, 2022.

[16] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *ICML*, 2021.

[17] Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *CVPR*, 2022.

[18] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, Seyedeh Sara Mahdavi, Raphael Gontijo Lopes, Tim Salimans, Jonathan Ho, David Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *NeurIPS*, 2022.

[19] Christoph Schuhmann, Romain Beaumont, Cade W Gordon, Ross Wightman, mehdi cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Richard Vencu, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5B: An open large-scale dataset for training next generation image-text models. *NeurIPS Datasets and Benchmarks*, 2022.

[20] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. LAION-400M: Open dataset of CLIP-filtered 400 million image-text pairs. *ArXiv*, 2021.

[21] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *ICLR*, 2021.

[22] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ICLR*, 2021.

[23] Xu Su, Jiaming Song, Chenlin Meng, and Stefano Ermon. Dual diffusion implicit bridges for image-to-image translation. *ArXiv*, 2022.

[24] Cristian Vaccari and Andrew Chadwick. Deepfakes and disinformation: Exploring the impact of synthetic political video on deception, uncertainty, and trust in news. *Social Media + Society*, 6, 2020.

[25] Zhou Wang, Alan Conrad Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13:600–612, 2004.

[26] Mika Westerlund. The emergence of Deepfake technology: A review. *Technology Innovation Management Review*, 9(11), 2019.

[27] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. GAN inversion: A survey. *ArXiv*, 2021.

[28] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Benton C. Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *ArXiv*, 2022.

[29] Min Zhao, Fan Bao, Chongxuan Li, and Jun Zhu. EGSDE: Unpaired image-to-image translation via energy-guided stochastic differential equations. *NeurIPS*, 2022.

(a) toys for boys (b) toys for girls

Figure 2: *(This figure is copied from [12])* GLIDE samples for the same random seed when changing the gender in the prompt. We can see that changes between corresponding image are minimal.

A Mathematical Details of Diffusion Models

A.1 DDPM and Non-Deterministic DDIM

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

$$\begin{aligned} &\boldsymbol{z} := \left(\boldsymbol{x}_T \oplus \boldsymbol{\epsilon}_T \oplus \cdots \oplus \boldsymbol{\epsilon}_2 \oplus \boldsymbol{\epsilon}_1\right) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \\ &\boldsymbol{x}_{T-1} = \boldsymbol{\mu}_T(\boldsymbol{x}_T, T) + \boldsymbol{\sigma}_T \odot \boldsymbol{\epsilon}_T, \\ &\boldsymbol{x}_{t-1} = \boldsymbol{\mu}_T(\boldsymbol{x}_t, t) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t, \quad T > t > 0, \\ &\boldsymbol{x} := \boldsymbol{x}_0. \end{aligned} \tag{5}$$

To be self-contained, here we provide details of $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t)$ and $\boldsymbol{\sigma}_t$ for DDPM [8] and DDIM [21]. 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 [8].

$$\boldsymbol{\mu}_T(\boldsymbol{x}_t, t) := \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right), \tag{6}$$

$$\boldsymbol{\sigma}_t := \begin{cases} \sqrt{\beta_t}\boldsymbol{I}, & \text{(option 1)} \\ \sqrt{\dfrac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}}\boldsymbol{I}, & \text{(option 2)} \\ \exp\left(\dfrac{\boldsymbol{v}_\theta(\boldsymbol{x}_t, t)}{2}\log\beta_t + \dfrac{\boldsymbol{I} - \boldsymbol{v}_\theta(\boldsymbol{x}_t, t)}{2}\log\dfrac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}\right). & \text{(option 3)} \end{cases} \tag{7}$$

DDIM's $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t)$ and $\boldsymbol{\sigma}_t$: We follow the notation in [21].

$$\boldsymbol{\mu}_T(\boldsymbol{x}_t, t) := \sqrt{\alpha_{t-1}}\left(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t), \tag{8}$$

$$\boldsymbol{\sigma}_t := \sigma_t\boldsymbol{I}, \qquad \text{where } \sigma_t = \eta\sqrt{(1-\alpha_{t-1})/(1-\alpha_t)}\sqrt{1-\alpha_t/\alpha_{t-1}}, \tag{9}$$

where η is a hyperparameter.

A.2 Score-based Generative Modeling with SDE

[22] 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 noises, we can readily incorporate their models and sampling methods into our framework. As a demonstration, here 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:

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}, t)\mathrm{d}t + \boldsymbol{\sigma}(t) \odot \mathrm{d}\boldsymbol{w}, \tag{10}$$

the reverse-time SDE is

$$\mathrm{d}\boldsymbol{x} = [\boldsymbol{f}(\boldsymbol{x}, t) - \boldsymbol{\sigma}(t)^2 \odot \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})]\mathrm{d}t + \boldsymbol{\sigma}(t) \odot \mathrm{d}\bar{\boldsymbol{w}}. \tag{11}$$

Suppose the forward SDE is discretized in the following form:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{f}_t(\boldsymbol{x}_t) + \boldsymbol{\sigma}_t \odot \boldsymbol{z}_t, \quad t = 0, \dots, T-1, \quad \boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{12}$$

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

$$\boldsymbol{x}_{t-1} = \boldsymbol{x}_t - \boldsymbol{f}_t(\boldsymbol{x}_t) + \boldsymbol{\sigma}_t^2 \odot \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t, \quad t = 1, \dots, T, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{13}$$

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. (5) and Eq. (13), we have $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t) := \boldsymbol{x}_t - \boldsymbol{f}_t(\boldsymbol{x}_t) + \boldsymbol{\sigma}_t^2 \odot \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$.

Algorithm 1: DPM-Encoder

Input: an image $\boldsymbol{x} := \boldsymbol{x}_0$, a pre-trained DDPM or non-deterministic DDIM with $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t)$, $\boldsymbol{\sigma}_t$, and forward distribution $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$

1. Sample $\boldsymbol{x}_1, \dots, \boldsymbol{x}_{T-1}, \boldsymbol{x}_T \sim q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$
2. $\boldsymbol{z} = \boldsymbol{x}_T$

for $t = T, \dots, 1$ **do**

3. $\boldsymbol{\epsilon}_t = \big(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_t, t)\big)/\boldsymbol{\sigma}_t$
4. $\boldsymbol{z} = \boldsymbol{z} \oplus \boldsymbol{\epsilon}_t$

5. **Return:** $\boldsymbol{z}$

Algorithm 2: DPM-Encoder for latent diffusion models (LDMs)

Input: an image $\boldsymbol{x}$, a pre-trained LDM, which consists of (1) an encoder E, (2) a decoder D, and (3) a "latent-space"[1] DDPM or non-deterministic DDIM $\boldsymbol{\mu}_T(\boldsymbol{z}_t, t)$, $\boldsymbol{\sigma}_t$, and forward distribution $q(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0)$

1. $\boldsymbol{z}_0 = E(\boldsymbol{x})$
2. Sample $\boldsymbol{z}_1, \dots, \boldsymbol{z}_{T-1}, \boldsymbol{z}_T \sim q(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0)$
3. $\boldsymbol{z} = \boldsymbol{z}_T$

for $t = T, \dots, 1$ **do**

4. $\boldsymbol{\epsilon}_t = \big(\boldsymbol{z}_{t-1} - \boldsymbol{\mu}_T(\boldsymbol{z}_t, t)\big)/\boldsymbol{\sigma}_t$
5. $\boldsymbol{z} = \boldsymbol{z} \oplus \boldsymbol{\epsilon}_t$

6. **Return:** $\boldsymbol{z}$

B Mathematical Details of DPM-Encoder

In this section, we provide details of our DPM-Encoder introduced in Section 2.2, which samples $\boldsymbol{z} \sim \mathrm{DPMEnc}(\boldsymbol{z}|\boldsymbol{x})$. For each image $\boldsymbol{x} := \boldsymbol{x}_0$, DDPM and its variants (e.g., non-deterministic DDIM) define a posterior distribution over the noisy images $\boldsymbol{x}_{1:T}$, denoted as $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ [8, 21]. 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 [8].

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) := \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}), \quad q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) := \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \mathbf{I}\right). \tag{14}$$

DDIM's posterior $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$: We follow the notation in [21].

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) := q(\boldsymbol{x}_T|\boldsymbol{x}_0) \prod_{t=2}^{T} q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0), \tag{15}$$

$$q(\boldsymbol{x}_T|\boldsymbol{x}_0) = \mathcal{N}(\sqrt{\alpha_T}\boldsymbol{x}_0, (1-\alpha_T)\,\boldsymbol{I}), \tag{16}$$

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}}, \sigma_t^2 \boldsymbol{I}\right), \tag{17}$$

$$\text{where } \sigma_t = \eta\sqrt{(1-\alpha_{t-1})\,/\,(1-\alpha_t)}\sqrt{1-\alpha_t/\alpha_{t-1}}.$$

Based on the posterior distribution $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$, DPM-Encoder samples 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. (5). Formally, we define the sampling process $\boldsymbol{z} \sim \mathrm{DPMEnc}(\boldsymbol{z}|\boldsymbol{x})$ as

$$\begin{gathered}\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T-1}, \boldsymbol{x}_T \sim q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0), \quad \boldsymbol{\epsilon}_t = \big(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_t, t)\big)/\boldsymbol{\sigma}_t, \quad t = T, \ldots, 1, \\ \boldsymbol{z} := \big(\boldsymbol{x}_T \oplus \boldsymbol{\epsilon}_T \oplus \cdots \oplus \boldsymbol{\epsilon}_2 \oplus \boldsymbol{\epsilon}_1\big).\end{gathered} \tag{18}$$

DPM-Encoder is guaranteed to be invertible. The proof is straightforward, provided as follows.

Proposition 1. *(Invertibility of DPM-Encoder) For each $\boldsymbol{z} \sim \mathrm{DPMEnc}(\boldsymbol{z}|\boldsymbol{x})$ defined in Eq. (18), we have $\boldsymbol{x} = \hat{\boldsymbol{x}}$, where $\hat{\boldsymbol{x}}$ is defined as*

$$\begin{aligned}&\hat{\boldsymbol{x}}_{T-1} = \boldsymbol{\mu}_T(\boldsymbol{x}_T, T) + \boldsymbol{\sigma}_T \odot \boldsymbol{\epsilon}_T, \\ &\hat{\boldsymbol{x}}_{t-1} = \boldsymbol{\mu}_T(\hat{\boldsymbol{x}}_t, t) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t, \quad T > t > 0, \\ &\hat{\boldsymbol{x}} := \hat{\boldsymbol{x}}_0.\end{aligned} \tag{19}$$

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

$$\begin{aligned}\hat{\boldsymbol{x}}_{T-1} &= \boldsymbol{\mu}_T(\boldsymbol{x}_T, T) + \boldsymbol{\sigma}_T \odot \boldsymbol{\epsilon}_T && (20)\\ &= \boldsymbol{\mu}_T(\boldsymbol{x}_T, T) + \boldsymbol{\sigma}_T \odot \big(\boldsymbol{x}_{T-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_T, T)\big)/\boldsymbol{\sigma}_T && (21)\\ &= \boldsymbol{\mu}_T(\boldsymbol{x}_T, T) + \big(\boldsymbol{x}_{T-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_T, T)\big) && (22)\\ &= \boldsymbol{x}_{T-1}. && (23)\end{aligned}$$

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

$$\begin{aligned}\hat{\boldsymbol{x}}_{t-1} &= \boldsymbol{\mu}_T(\hat{\boldsymbol{x}}_t, t) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t && (24)\\ &= \boldsymbol{\mu}_T(\boldsymbol{x}_t, t) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t && (25)\\ &= \boldsymbol{\mu}_T(\boldsymbol{x}_t, t) + \boldsymbol{\sigma}_t \odot \big(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_t, t)\big)/\boldsymbol{\sigma}_t && (26)\\ &= \boldsymbol{\mu}_T(\boldsymbol{x}_t, t) + \big(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_t, t)\big) && (27)\\ &= \boldsymbol{x}_{t-1}. && (28)\end{aligned}$$

□

DPM-Encoder is also applicable to latent diffusion models (LDMs), shown in Algorithm 2. Specifically, we first use the LDM encoder E to encode $\boldsymbol{z}_0 = E(\boldsymbol{x})$ and sample $\boldsymbol{z} \sim \mathrm{DPMEnc}(\boldsymbol{z}|\boldsymbol{z}_0; G_{\text{latent}})$ with the diffusion model G_{latent} used to model the "latent space"[1] distribution over $\boldsymbol{z}_0$. When applied to LDMs, the invertibility over the image $\boldsymbol{x}$ depends on the LDM encoder E and LDM decoder D.

C CycleDiffusion for Unpaired Image-to-Image Translation

Suppose we have two stochastic DPMs G_1 and G_2 trained on similar domains $\mathcal{D}_1$ and $\mathcal{D}_2$ (e.g., cats and dogs) independently, and the two models have the same noise schedule $\boldsymbol{\sigma}_t$ but not necessarily the same architecture. We find that the two models naturally learn a common latent space. In other words, for each latent code $\boldsymbol{z}$, $G_1(\boldsymbol{z})$ and $G_2(\boldsymbol{z})$ have common details. One possible explanation is that, in Eq. (1), $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t)$ models the patterns averaged in one domain, while $\boldsymbol{\epsilon}_t$ models details specific to one image. Therefore, $G_1(\boldsymbol{z})$ and $G_2(\boldsymbol{z})$ have domain-specific patterns but common details. Based on this finding, our **CycleDiffusion** is also applicable to unpaired image-to-image translation. Given a source image $\boldsymbol{x} \in \mathcal{D}_1$, we encode it as $\boldsymbol{z}$ and then decode it as $\hat{\boldsymbol{x}} = G_2(\boldsymbol{z})$:

$$\boldsymbol{z} \sim \mathrm{DPMEnc}(\boldsymbol{z}|\boldsymbol{x}, G_1), \quad \hat{\boldsymbol{x}} = G_2(\boldsymbol{z}). \tag{29}$$

D Variants of CycleDiffusion

Algorithm 3 presents the pseudo-code for our original formulation of CycleDiffusion (for unpaired image-to-image translation between two domains) in Eq. (29), and Algorithm 4 presents the pseudo-code for our original formulation of CycleDiffusion (for zero-shot image-to-image translation with text-to-image diffusion models) in Eq. (3). In the following, we will introduce two improvements

of CycleDiffusion that can further improve the performance. The first one is to use SDEdit [11] for noise removal. The second one is to use early stopping to improve content preservation. Although we used them in Section 3 and Appendix E separately, it should be noted that the two improvements are orthogonal and can be used together.

Algorithm 3: CycleDiffusion for unpaired image-to-image translation (original version)

Input: an image $\boldsymbol{x} := \boldsymbol{x}_0$; source-domain stochastic DPM G_1, whose mean estimator is $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t; \boldsymbol{\theta}_1)$; target-domain stochastic DPM G_2, whose mean estimator is $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t; \boldsymbol{\theta}_2)$; the covariance $\boldsymbol{\sigma}_t$ and posterior distribution $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ shared by the two DPMs

1. Sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T-1}, \boldsymbol{x}_T \sim q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$

for $t = T, \ldots, 1$ **do**

  2. $\boldsymbol{\epsilon}_t = \big(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_t, t; \boldsymbol{\theta}_1)\big)/\boldsymbol{\sigma}_t$

3. $\hat{\boldsymbol{x}}_T = \boldsymbol{x}_T$

for $t = T, \ldots, 1$ **do**

  4. $\hat{\boldsymbol{x}}_{t-1} = \boldsymbol{\mu}_T(\hat{\boldsymbol{x}}_t, t; \boldsymbol{\theta}_2) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t$

5. **Return:** $\hat{\boldsymbol{x}} := \hat{\boldsymbol{x}}_0$

Algorithm 4: CycleDiffusion for zero-shot image-to-image translation (original version)

Input: a source image $\boldsymbol{x} := \boldsymbol{x}_0$; a source text $\boldsymbol{t}$; a target text $\hat{\boldsymbol{t}}$; a pre-trained text-to-image stochastic DPM G, whose mean estimator is $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t|\boldsymbol{t})$, where $\boldsymbol{t}$ is the text condition; the covariance $\boldsymbol{\sigma}_t$ and posterior distribution $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$

1. Sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T-1}, \boldsymbol{x}_T \sim q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$

for $t = T, \ldots, 1$ **do**

  2. $\boldsymbol{\epsilon}_t = \big(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_t, t|\boldsymbol{t})\big)/\boldsymbol{\sigma}_t$

3. $\hat{\boldsymbol{x}}_T = \boldsymbol{x}_T$

for $t = T, \ldots, 1$ **do**

  4. $\hat{\boldsymbol{x}}_{t-1} = \boldsymbol{\mu}_T(\hat{\boldsymbol{x}}_t, t|\hat{\boldsymbol{t}}) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t$

5. **Return:** $\hat{\boldsymbol{x}} := \hat{\boldsymbol{x}}_0$

D.1 CycleDiffusion with SDEdit Noise Removal

We observed that CycleDiffusion sometimes generates noisy outputs, and we use SDEdit [11] for noise removal. Specifically, we modify the original CycleDiffusion in Eq. (29) as

$$\boldsymbol{z} \sim \mathrm{DPMEnc}(\boldsymbol{z}|\boldsymbol{x}, G_1), \quad \tilde{\boldsymbol{x}} = G_2(\boldsymbol{z}), \quad \hat{\boldsymbol{x}} = \mathrm{SDEdit}(\tilde{\boldsymbol{x}}, T_{\text{sdedit}}), \tag{30}$$

where $\mathrm{SDEdit}(\tilde{\boldsymbol{x}}, T_{\text{sdedit}})$ means that we apply SDEdit to $\tilde{\boldsymbol{x}}$ for T_{sdedit} steps with a DDPM trained on the target domain. When $T = 1000$, we find that $T_{\text{sdedit}} \approx 100$ gives us good noise removal.

Application in the experiments: In Appendix E, we used $T = 1000$, and we set $T_{\text{sdedit}} = 100$ for the Cat $\rightarrow$ Dog direction and $T_{\text{sdedit}} = 125$ for the Wild $\rightarrow$ Dog direction. In Section 3, we did not use this noise removal since latent diffusion models seem not to generate noisy outputs.

D.2 CycleDiffusion with Early Stopping

In this part, we propose another improvement of CycleDiffusion using the idea of early stopping to achieve better content preservation in image-to-image translation. Algorithm 5 presents the modified pseudo-code for unpaired image-to-image translation between two domains. Algorithm 6 presents the modified pseudo-code for zero-shot image-to-image translation with text-to-image diffusion models.

Application in the experiments: In Appendix E, we did not use this early stopping since the current CycleDiffusion with noise removal (Appendix D.1) produces satisfactory results. In Section 3, we used this early stopping idea to achieve better content preservation.

Algorithm 5: CycleDiffusion for unpaired image-to-image translation (early stopping)

Input: an image $\boldsymbol{x} := \boldsymbol{x}_0$; source-domain stochastic DPM G_1, whose mean estimator is $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t; \boldsymbol{\theta}_1)$; target-domain stochastic DPM G_2, whose mean estimator is $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t; \boldsymbol{\theta}_2)$; the covariance $\boldsymbol{\sigma}_t$ and posterior distribution $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ shared by the two DPMs; early stopping step $T_{\text{es}} < T$

1. Sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T-1}, \boldsymbol{x}_T \sim q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$

for $t = T_{es}, \ldots, 1$ **do**

  2. $\boldsymbol{\epsilon}_t = \big(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_t, t; \boldsymbol{\theta}_1)\big)/\boldsymbol{\sigma}_t$

3. $\hat{\boldsymbol{x}}_{T_{\text{es}}} = \boldsymbol{x}_{T_{\text{es}}}$

for $t = T_{es}, \ldots, 1$ **do**

  4. $\hat{\boldsymbol{x}}_{t-1} = \boldsymbol{\mu}_T(\hat{\boldsymbol{x}}_t, t; \boldsymbol{\theta}_2) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t$

5. **Return:** $\hat{\boldsymbol{x}} := \hat{\boldsymbol{x}}_0$

Algorithm 6: CycleDiffusion for zero-shot image-to-image translation (early stopping)

Input: a source image $\boldsymbol{x} := \boldsymbol{x}_0$; a source text $\boldsymbol{t}$; a target text $\hat{\boldsymbol{t}}$; a pre-trained text-to-image stochastic DPM G, whose mean estimator is $\boldsymbol{\mu}_T(\boldsymbol{x}_t, t|\boldsymbol{t})$, where $\boldsymbol{t}$ is the text condition; the covariance $\boldsymbol{\sigma}_t$ and posterior distribution $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$; early stopping step $T_{\text{es}} < T$

1. Sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T-1}, \boldsymbol{x}_T \sim q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$

for $t = T_{es}, \ldots, 1$ **do**

  2. $\boldsymbol{\epsilon}_t = \big(\boldsymbol{x}_{t-1} - \boldsymbol{\mu}_T(\boldsymbol{x}_t, t|\boldsymbol{t})\big)/\boldsymbol{\sigma}_t$

3. $\hat{\boldsymbol{x}}_{T_{\text{es}}} = \boldsymbol{x}_{T_{\text{es}}}$

for $t = T_{es}, \ldots, 1$ **do**

  4. $\hat{\boldsymbol{x}}_{t-1} = \boldsymbol{\mu}_T(\hat{\boldsymbol{x}}_t, t|\hat{\boldsymbol{t}}) + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t$

5. **Return:** $\hat{\boldsymbol{x}} := \hat{\boldsymbol{x}}_0$

Table 2: Quantitative comparison for unpaired image-to-image translation methods. Methods in the second block use the same pre-trained diffusion model in the target domain. Results of CUT, ILVR, SDEdit, and EGSDE are from [29]. Best results using diffusion models are in **bold**. CycleDiffusion has the best FID and KID among all methods, and the best SSIM among methods with diffusion models. Note that it has been shown that SSIM is much better correlated with human visual perception than squared distance-based metrics such as L_2 and PSNR [25].

	Cat → Dog				Wild → Dog			
	FID↓	KID×10^3↓	PSNR↑	SSIM↑	FID↓	KID×10^3↓	PSNR↑	SSIM↑
CUT (GAN SOTA; [13])	76.21	–	17.48	0.601	92.94	–	17.20	0.592
ILVR [2]	74.37	–	17.77	0.363	75.33	–	16.85	0.287
SDEdit [11]	74.17	–	19.19	0.423	68.51	–	17.98	0.343
EGSDE [29]	65.82	–	**19.31**	0.415	59.75	–	**18.14**	0.343
DDIB [23]	352.15	320.6	7.234	0.123	307.05	225.6	8.60	0.164
CycleDiffusion w/ DDIM ($\eta = 0.1$)	**55.93**	**18.6**	16.78	**0.537**	**54.26**	**16.9**	16.47	**0.449**

E Experiments on Unpaired Image-to-Image Translation

Given two unaligned image domains, unpaired image-to-image translation aims at mapping images in one domain to the other. We follow setups from previous works whenever possible, as detailed below.

Baselines: We compared CycleDiffusion with previous unpaired image-to-image translation methods: CUT [13], ILVR [2], SDEdit [11], and EGSDE [29]. We tried to implement DDIB [23] for this task but failed to produce reasonable results, so we omitted this baseline in this experiment.

Datasets: Following previous work [13, 29], we conducted experiments on the test set of AFHQ [3] with resolution 256×256. Two directions were considered: Cat → Dog and Wild → Dog.

Metrics: To evaluate the realism, we reported Frechet Inception Distance (FID; [7]) and Kernel Inception Distance (KID; [1]) between the set of translated images and the set of target images. To

evaluate the faithfulness, we reported PSNR and SSIM [25] between each translated image and its source image, averaged across all instances.

Pre-trained diffusion models: ILVR, SDEdit, and EGSDE only need the diffusion model trained on the target domain (i.e., Dog). We followed them to use the pre-trained model from [2] for Dog. CycleDiffusion needs diffusion models on both domains, so we trained the Cat and Wild models.

Seen in Table 2 are the results. CycleDiffusion has the best realism (i.e., FID and KID). There is a mismatch between the faithfulness metrics (i.e., PSNR and SSIM), and note that SSIM is much better correlated with human perception than PSNR [25]. Among all diffusion model-based methods, CycleDiffusion achieves the highest SSIM. Figure 3 displays some image samples.

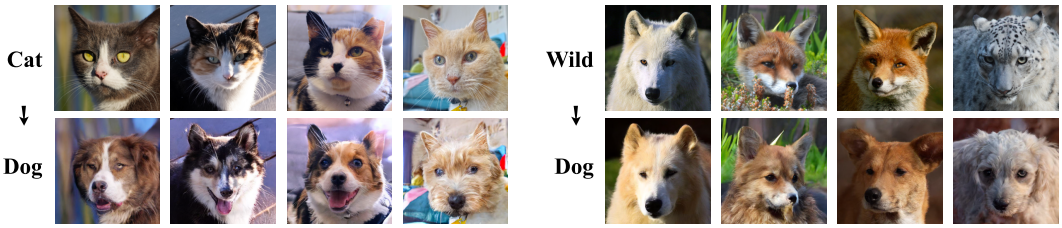

Figure 3: Unpaired image-to-image translation (Cat → Dog, Wild → Dog, and FFHQ → CelebAHQ).

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

CycleDiffusion and the baselines for zero-shot image-to-image translation have some shared hyperparameters; each method also has its own unique hyperparameters. For both CycleDiffusion and the baselines, we can enumerate some of these parameters and select the best per sample based on a certain criterion. Moreover, for stochastic methods such as SDEdit and our CycleDiffusion, one may run several trials. In this section, we provide these details.

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: Since DDIB only applies to deterministic DPMs, 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: For SDEdit, 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 SDEdit 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 step 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.

G Additional Results for Zero-Shot Image-to-Image Translation

Figure 4 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.

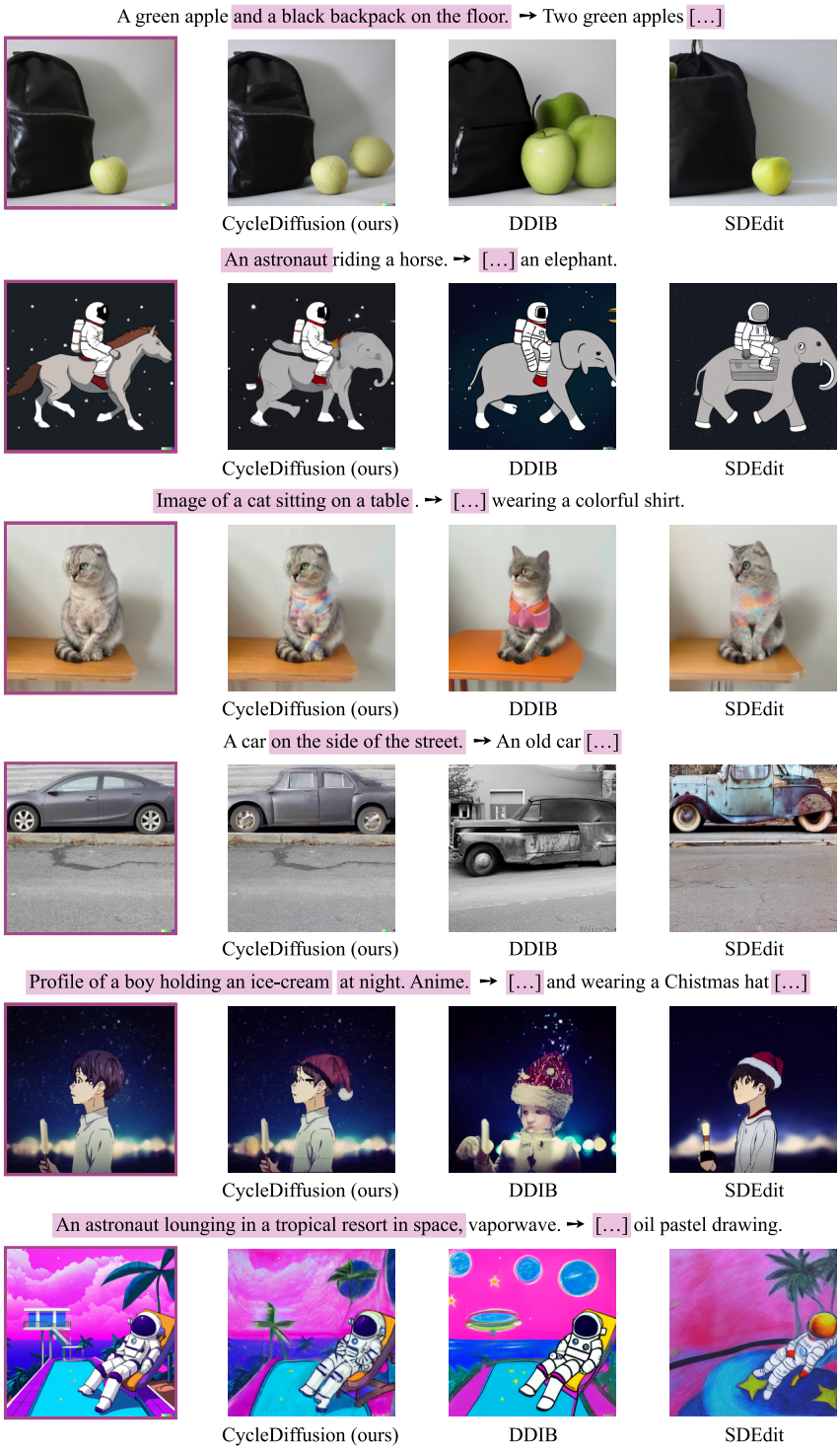

Figure 4: Samples for zero-shot image-to-image translation. Notations follow Figure 1. Compared with DDIB and SDEdit, CycleDiffusion greatly improves the faithfulness to the source image.

H Societal Impact

Improved generative modeling usually leads to improved accessibility of creating deceptive media (e.g., DeepFakes [26, 24]) and privacy leaks (e.g., information leaks from large-scale pre-training data of text-to-image diffusion models). Moreover, when applying CycleDiffusion to text-to-image diffusion models, one may get some zero-shot edited images that reflect the unintended biases learned by the text-to-image diffusion models. As a remedy, we note that several existing technologies can detect generated images effectively, and others can handle privacy leaks during pre-training. We encourage researchers and practitioners to consider the aforementioned risks and remedies when deploying systems using methods developed in this paper.