FlowGrad: Controlling the Output of Generative ODEs with Gradients

Xingchao Liu[1], Lemeng Wu[1], Shujian Zhang[1], Chengyue Gong[1], Wei Ping[2], Qiang Liu[1]
[1]University of Texas at Austin [2]NVIDIA
{xcliu,lmwu,szhang19,cygong}@utexas.edu,wping@nvidia.com,lqiang@cs.utexas.edu

Abstract

Generative modeling with ordinary differential equations (ODEs) has achieved fantastic results on a variety of applications. Yet, few works have focused on controlling the generated content of a pre-trained ODE-based generative model. In this paper, we propose to optimize the output of ODE models according to a guidance function to achieve controllable generation. We point out that, the gradients can be efficiently back-propagated from the output to any intermediate time steps on the ODE trajectory, by decomposing the back-propagation and computing vector-Jacobian products. To further accelerate the computation of the back-propagation, we propose to use a non-uniform discretization to approximate the ODE trajectory, where we measure how straight the trajectory is and gather the straight parts into one discretization step. This allows us to save $\sim$ 90% *of the back-propagation time with ignorable error. Our framework, named FlowGrad, outperforms the state-of-the-art baselines on text-guided image manipulation. Moreover, FlowGrad enables us to find global semantic directions in* frozen *ODE-based generative models that can be used to manipulate new images without extra optimization.*

1. Introduction

Controllable generation is very important for image editing [2, 11, 43], text-guided image manipulation [19, 30, 33], etc.. Traditionally, we use GAN and optimize the latent embedding with the desirable objective functions [2, 7, 11, 29, 31, 38, 49]. But the disadvantage is that, it is difficult to embed the image into the GAN space honestly and the performance is limited by the pre-trained GANs, which suffer from training instability and mode collapse.

Recently, diffusion models (or stochastic differential equation (SDE)-based generative models) has been popular [9, 12, 30, 39, 40, 42, 45], and there has been a number of works on controlled generation based on diffusion models, such as [9, 28]. But due to the diffusion noise, it is hard to accurately control the output, especially when it comes to optimizing a complex loss function such as CLIP [35]. To achieve controlled generation, existing methods either requires training a noised version of the guidance [9, 30, 40], or fine-tuning the whole diffusion model [13, 19].

In contrast, ordinary differential equation (ODE)-based generative models represent a simpler alternative than diffusion without involving the diffusion noise and the Ito calculus machinery. Recently, it has been shown that 1) ODEs can be trained directly without resorting to SDEs, and 2) ODE can perform comparable or even better than SDE [16, 21, 36, 44].

Due to the deterministic nature of ODEs, they form an ideal model for controlled generation, as they enjoy both the rich latent space as SDEs and the explicit optimization framework as GANs. The goal of this work is to fully explore its potential in terms of controlled generation, with unconditioned pre-trained ODEs. Technically, 1) We present a simple way to control the output of ODE-based deep generative models with gradients; 2) We present a novel strategy to speedup the gradient computation by explore the straightness of ODE trajectories. By measuring the straightness at each time step during the simulation with Euler discretization, we can approximate the ODE trajectory with a few-step non-uniform discretization, and consequently reduce a great amount of time in back-propagation.

Our fast gradient computation scheme, named FlowGrad, allows us to efficiently control the generated contents of ODEs with any differentiable loss functions. In particular, we test FlowGrad on a challenging objective function, the CLIP loss, to manipulate user-provided images with text prompts. Moreover, by optimizing a set of training images together, FlowGrad can find semantically meaningful global directions in pre-trained ODE models, which allow manipulating new images for free. Equipped with advanced ODE-based generative models, FlowGrad outperforms state-of-the-art CLIP-guided diffusion models and GANs.

2. Background

In this section, we introduce background knowledge of ODE-based generative models.

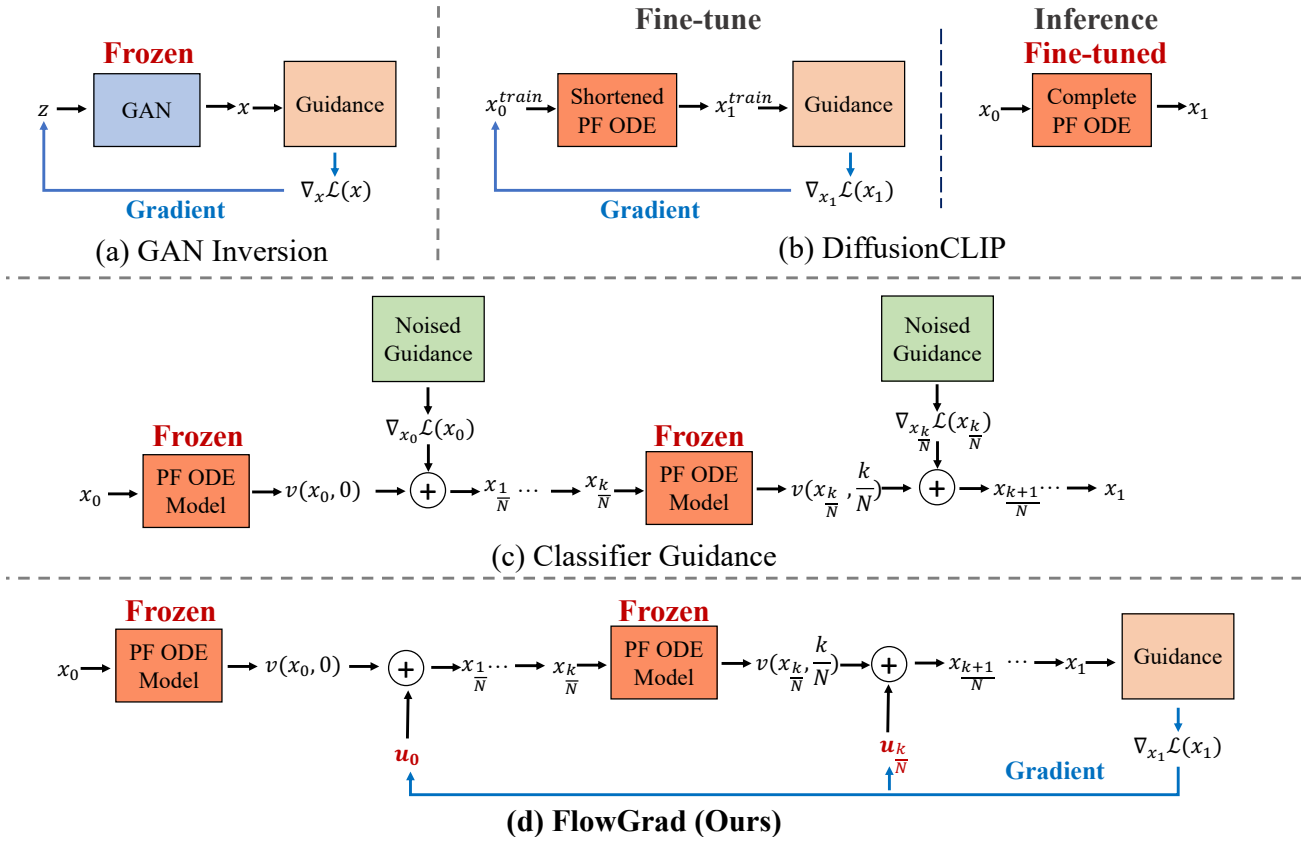


Figure 1. We compare different frameworks for controlled generation with generative models and a guidance objective $\mathcal{L}$. (a) GAN can be directly used as a prior by optimizing its latent z. However, a good z for initialization can be hard to get [1, 2, 22]. (b) DiffusionCLIP [19] fine-tunes a pre-trained ODE model to minimize the guidance loss. To allow back-propagation during fine-tuning, it adopts a shortened ODE with large discretization error. When applied on a new image, its performance is bounded by both the generalization error and the discretization error. (c) Classifier guidance [13] can produce high-quality samples, but it needs to train a new guidance model that provides gradient for the noised version of the images in every intermediate step. (d) Our FlowGrad adds control variables to each time step of the complete ODE trajectory, and optimizes the control variables with gradients from the guidance model. Compared with other methods, FlowGrad has smaller discretization error, enjoys the powerful encoding ability of ODE models, and does not require a noised guidance model.

2.1. Probability Flows

Given i.i.d. samples $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$ from a data distribution π_{data}, a probability flow can be effectively learned by training a velocity field $v_\theta(x_t, t)$ of an ordinary differential equation, which is indexed by a continuous time variable $t \in [0, T]$, to map a simple distribution π_0 to the complex target distribution π_{data}. Without loss of generality, we assume $T = 1$. Probability flows shift the initial distribution π_0 in a deterministic way,

$$\mathrm{d}x_t = v_\theta(x_t, t)\mathrm{d}t, \quad x_0 \sim \pi_0, \tag{1}$$

Traditionally, v_θ can be trained by the adjoint method [10] with the extra cost of solving an ODE during training. Later, researchers propose to extract probability flows from learned diffusion models as a way to save computation [39, 40]. Recently, a series of methods focus on directly designing the target velocity field and learning v_θ by regressing the target field [3, 21, 44].

Once v_θ is learned, one can simply sample from the ODE to get x_1 as draws from the target distribution π_1 with N-step Euler solver,

$$x_{(k+1)/N} = x_{k/N} + \frac{1}{N} v_\theta(x_{k/N}, k/N), \tag{2}$$

where $k \in \{0, 1, \ldots, N-1\}$, $x_0 = x_{0/N}$ is a random sample from π_0 and $x_1 = x_{N/N}$ is the generated data. The number of discretization steps, N, determines the closeness of the simulated trajectory (2) and the continuous ODE trajectory (1). When $N \to \infty$, the simulated trajectory (2) has approaches the same end point as the continuous one. In practice, it is often found that an appropriate choice of N for existing probability flow ODEs ranges from 20 to 200.

Representative ODE-based Generative Models Exisitng ODE-based generative models can be can be characterized by the same framework [23, 40, 42]. Let X_t be any differentiable interpolation of $x_0 \sim \pi_0$ and $x_1 \sim \pi_{data}$, then, v can be learned by minimizing

$$\mathbb{E}_{\substack{t\sim\text{Uni}(0,1)\\ x_0\sim\pi_0\\ x_1\sim\pi_{data}}}[\left\|\dot{X}_t - v_\theta(X_t,t)\right\|^2], \tag{3}$$

where $\dot{X}_t$ denotes the time derivative of X_t. Rectified flow [21] is the special case when $X_t = tx_1 + (1-t)x_0$, and DDIM [39, 40] corresponds to $X_t = \alpha_t x_1 + \beta_t x_0$ with x_0 a Gaussian distribution, $\alpha_t = \exp\left(-5a(1-t)^2 - 0.05(1-t)\right)$ and $\beta_t = \sqrt{1-\alpha_t^2}$.

Straightness and Fast Simulation The N needed to accurately discretize and simulate the ODE depends on how straight the trajectories are. In particular, if the trajectories of the ODE is straight, that is $v_\theta(X_t,t) = v_\theta(x_0,0), \forall t \in [0,1]$, where $X_t = tx_1 + (1-t)x_0$ and x_1 is the end point of the trajectory, then a single Euler step yields exact solution,

$$x_1 = x_0 + v_\theta(x_0,0). \tag{4}$$

Most of the existing ODEs do not necessarily produce straight trajectories. Although [21] proposes a *reflow* procedure to straighten the ODE trajectory and allow generation with very small N, it still needs a curved trajectory to generate high-quality images. For these typical non-straight ODE trajectories, a small N leads to non-realistic blurry generation.

2.2. Encoding and Latent Space

The ODE model can be viewed as an auto-encoder, in which each data point x_1 can be encoded to x_0, and the decoding ($x_0 \to x_1$) and encoding ($x_1 \to x_0$) can be simply realized by solving the ODE forwardly, and backwardly. Specifically, with N-step Euler solver, we can encode x_1 by

$$x_{k/N} = x_{(k+1)/N} - \frac{1}{N} v_\theta(x_{(k+1)/N}, (k+1)/N), \tag{5}$$

Encoding/Decoding with ODEs is extremely simple with nearly perfect reconstruction. This avoids the complicated encoding strategy and imperfect reconstruction in GAN inversion [1, 2, 41], making ODEs favorable for real image editing.

3. FlowGrad

In this section, we introduce how to efficiently back-propagate the gradient through the trajectory of an ODE, and provide acceleration schemes to effectively decrease the computational cost.

3.1. The Optimal Control Framework and Challenges

Beyond unconditioned generation, we would like to directly control the generated contents given by the ODE from a starting point x_0. To achieve this, we can define a differentiable loss function $\mathcal{L}$ to control the end point x_1 towards our expectation by minimizing the following optimal control problem,

$$\begin{aligned} \min_u \quad & \mathcal{L}(x_1) + \lambda \int_0^1 ||u(t)||^2 \mathrm{d}t, \\ s.t. \quad & x_1 = x_0 + \int_0^1 \left(v_\theta(x_t,t) + u(t)\right) \mathrm{d}t, \end{aligned} \tag{6}$$

where $u(t)$ is a control function. This continuous optimal control problem is intractable, since the velocity field v_θ, which is usually represented by a neural network, is complicated. Instead, we turn to the discretized version of this problem,

$$\begin{aligned} \min_u \quad & \mathcal{J} := \mathcal{L}(x_1) + \lambda \sum_{k=0}^{N-1} ||u_{k/N}||^2 \\ s.t. \quad & x_{(k+1)/N} = x_{k/N} + \frac{1}{N}\left(v_\theta(x_{k/N}, k/N) + u_{k/N}\right), \end{aligned} \tag{7}$$

where $k \in \{0,1,\ldots,N-1\}$ and we adopt N-step Euler discretization.

Challenges A straightforward optimization method for problem (7) is to run the simulation with Euler discretization, compute the gradient for each $u_{k/N}$ and perform gradient descent. However, compute the gradient $\nabla_{u_{k/N}}\mathcal{J}$ needs to back-propagate through a chain with $N-k$ nested v_θ. Since v_θ is a neural network, computing the gradient for the nested structure requires overwhelming GPU memory in current auto-differentiation framework, like PyTorch [32], when the depth is large. For example, with a Nvidia TITAN XP GPU with 12GB memories, we can only compute the gradients for $k < 4$ with direct back-propagation.

3.2. Decomposition of the Back-propagation

To allow gradient computation for all the $u_{k/N}$, we propose to decompose the gradient computation by leveraging the following fact,

$$\nabla_{x_{k/N}}\mathcal{J} = (\nabla_{x_{(k+1)/N}}\mathcal{J}) \cdot J_{\phi_k}(x_{k/N}), \tag{8}$$

where $\phi_k(x) = x + \frac{1}{N}(v_\theta(x, k/N) + u_{k/N})$ and $J_{\phi_k}(x_{k/N})$ is the Jacobian of ϕ_k at point $x_{k/N}$. Typically, the Jacobian is costly to obtain in the auto-differentiation framework. Fortunately, Eq. (8) is a vector-Jacobian product, which is much easier to get than the whole Jacobian. For

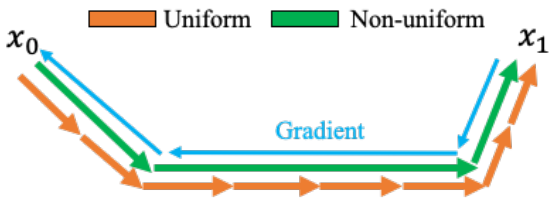


Figure 2. To accelerate the back-propagation, we approximate the trajectory generated by Euler discretization using non-uniform discretization. We can significantly decrease the number of vector-Jacobian products with acceptable error in gradient computation.

example, in PyTorch, the vector-Jacobian product can be efficiently computed with the double-backwards trick [32], at the cost of two normal backwards. After the gradient at the end point, $\nabla_{x_1}\mathcal{J}(x_1)$, is computed, we can get the gradient $\nabla_{u_{k/N}}\mathcal{J} = \frac{1}{N}\nabla_{x_{(k+1)/N}}\mathcal{J}$ by iteratively applying Eq. (8). Then we can then minimize $\mathcal{J}$ with gradient descent,

$$u_{k/N} \leftarrow u_{k/N} - \alpha\nabla_{u_{k/N}}\mathcal{J}, \tag{9}$$

where α is the step size set by the user.

3.3. Acceleration: Using non-uniform discretization

Although in Section 3.2, we are able to solve the optimization problem (7) through decomposing the back-propagation, its time consumption is still unacceptable. To compute the gradients for all the $u(k/N)$ and perform one gradient descent, it takes $\sim 18s$ on TITAN XP with a pre-trained Rectified Flow model on 256×256 images when $N = 100$. In comparison, it only takes $7s$ to simulate the ODE trajectory and generate the image x_1 with Euler discretization.

For better efficiency in back-propagation, we turn to non-uniform discretization rather than naive Euler discretization. Non-uniform discretization simulates the ODE with the following equation,

$$x_{t_{j+1}} = x_{t_j} + (t_{j+1} - t_j)v_\theta(x_{t_j}, t_j), \tag{10}$$

where $G := \{t_j\}_{j=0}^{N'}$ is a set of grid points that satisfies $t_0 = 0$ and $t_{N'} = 1$. If we can find G that contains only a few elements but small discretization error, then we can significantly reduce the number of applying Eq. (8) and hereby save time.

Inspired by the properties of straight ODEs, we propose to *detect* the straight parts on the ODE trajectory and approximate the trajectory with the non-uniform discretization for fast back-propagation. Formally, our method consists of three steps:

Method	Full BP	$\xi = 5e-3$	$\xi = 1e-2$
Time (s)	17.7	1.4	1.1

Table 1. Average running time for computing the gradients for all the variables on the trajectory with a pre-trained Rectified Flow ODE and the same TITAN XP GPU.

Step 1: ODE Simulation We run Euler discretization for N steps to generate the ODE trajectory as in Section 3.1 and save the velocities $\{v_k := v_\theta(x_{k/N}, k/N)\}_{k=0}^{N-1}$.

Step 2: Approximation with Non-uniform Discretization We define the straightness at time k/N as,

$$S(k/N) = \max(d(v_{k-1}, v_k), d(v_{k+1}, v_k)), \tag{11}$$

where $d(v_1, v_2) = ||v_1 - v_2||^2/||v_2||^2$ measures the relative change in velocity. If $S(k/N)$ is small, the velocity is almost invariant at time k/N and we can consider $t \in [(k-1)/N, (k+1)/N]$ is 'straight'. Using the straightness measurement, G can be constructed by,

$$t_0 = 0,$$
$$N(t_{j+1} - t_j) = \arg\min_m \left\{ \sum_{i=1}^{m} S(t_j + \frac{i}{N}) \geq \xi, \right\}, \tag{12}$$
where $1 \leq m \leq N(1 - t_j)$ and stop when $t_j = 1$.

Here, ξ is a hyper-parameter to control the tolerance of straightness. As ξ increases, the minimal m becomes larger and the number of elements in G decreases. Thus, a large ξ can reduce the computational time at the cost of introducing more error. Note that all the element t_j in G also belongs to the grid points of Euler discretization $\{k/N\}_{k=0}^{N}$.

Step 3: Fast Back-propagation Now that the ODE trajectory can be represented by the new non-uniform discretization with only $N' << N$ steps, we can compute the gradient for u_{t_j} with only N' times of vector-Jacobian product computation. Specifically, we have,

$$\nabla_{x_{t_j}}\mathcal{J} = (\nabla_{x_{t_{j+1}}}\mathcal{J}) \cdot J_{\phi'_j}(x_{t_j}), \tag{13}$$

where $\phi'_j(x) = x + (t_{j+1} - t_j)(v_\theta(x, t_j) + u_{t_j})$. And correspondingly, $\nabla_{u_{t_j}}\mathcal{J} = (t_{j+1} - t_j)\nabla_{x_{t_{j+1}}}\mathcal{J}$.

Re-assignment After the computation of $\{u_{t_j}\}_{j=1}^{N'}$, we re-assign the values to each $u_{k/N}$ to allow Euler discretization in the next iteration. Specifically, for $k \in [Nt_j, Nt_{j+1})$, we assign $u_{k/N} = u_{t_j}$, so that the result still holds when simulated with Euler discretization.

Algorithm 1 FlowGrad: An efficient algorithm for back-propagation and optimization with ODEs

Input: A pre-trained ODE velocity v_θ, a differentiable guidance for clean images $\mathcal{L}$, number of Euler discretization steps N, the number of optimization iterations M, penalty coefficient λ, threshold ξ and step size α.

Procedure:
Initialize all the variables $\{u(k/N)\}_{k=0}^{N-1}$ to zero.
for i in 1 : M **do**
  // Simulate the PF ODE trajectory
  for k in 0 : (N-1) **do**
    $x_{(k+1)/N} = x_{k/N} + \frac{1}{N}\left(v_\theta(x_{k/N}, k/N) + u(k/N)\right)$,
  end for
  Save $\{v_k := v_\theta(x_{k/N}, k/N)\}_{k=0}^{N-1}$.

  // Approximate with Non-uniform Discretization
  for k in 0 : (N-1) **do**
    $S(k/N) = \max(d(v_{k-1}, v_k), d(v_{k+1}, v_k))$,
  end for
  Construct G with Eq. (12).
  Compute gradient with Eq. (13).

  // Update the variables
  Compute $\{u_{t_j}\}_{j=0}^{N'}$ and re-assign the values to $u_{k/N}$.
end for

4. Related Works

GAN Inversion Using the latent space of pre-trained GANs for image editing has attracted much attention and achieved impressive results [43]. The rich latent spaces learned by modern GANs allow high-quality modification on images of human faces, animals, natural scenes, etc. [2,7,11,29,31,38,49], even generate out-of-distribution images under language guidance [22, 33]. However, due to the highly non-convex landscape of the latent space of pre-trained GANs, how to satisfactorily embed the images into the latent spaces is a difficult problem [1, 15, 43, 48], which affects the later performance on editing the images.

SDE and ODE-based Generative Models Recently, diffusion models gains popularity because of their astonishing power in learning generative models on various domains, e.g., images, molecules and point clouds [9, 12, 20, 23, 25, 37, 40, 42]. Unlike GANs, diffusion models do not have a one-to-one mapping between images and noises in the latent space. On the contrary, ODE-based models, either derived from pre-trained diffusion models [39, 40] or directly learned from scratch [16, 21, 44], can map a latent code to a determinisitc image, while enjoying the high generation quality like diffusion models. Moreover, by running the ODE solver in the reverse way, it is straightforward to find the corresponding latent code for a given image, with nearly perfect reconstruction, which makes it favorable over GAN inversion [40]. However, even equipped with acceleration methods like [4, 26], both SDE and ODE-based models require more than 20 iterations to generate a high-quality image, making it impossible to naively migrate the optimization techniques for image editing in GAN inversion to these novel models.

Controlled Generation with ODE and SDE-based Models Because of the inherent difference between GANs and SDE/ODE-based models, a variety of new methods have been proposed for controlled generation with diffusion models [5,6,18,19,27,28,30,34,40,47]. In [9,30,40], a guidance model for noisy images is trained to guide the model in the intermediate time steps of SDE. Training the noised guidance model is time-consuming and expensive. [5, 28] propose to edit or generate realistic images from coarse strokes or reference images, but their methods can only find images that are close in the Euclidean space thanks to the noise in SDEs. DiffusionCLIP [19] can manipulate images according to CLIP guidance with SDEs, but it needs to fine-tune the whole score function model for each text prompt. This causes problem in storage and time consumption. Our method can directly use guidance models pre-trained on clean images, and manipulate the images following text instructions with pre-trained *frozen* ODE models.

5. Experiments

We conduct a series of experiments on text-guided image manipulation to demonstrate the effectiveness of our FlowGrad. We apply FlowGrad on state-of-the-art ODE-based generative models, including Rectified Flow (RF), and Latent Diffusion Model (LDM) with DDIM. For all the experiments, we use the accelerated FlowGrad with $\xi = 5e-3$ without further notification. Code is available at https://github.com/gnobitab/FlowGrad.

5.1. Text-guided Image Manipulation

Zero-shot image manipulation with text guidance is a desirable application for generative models after the emergence of Contrastive Language-Image Pretraining (CLIP) [35]. In this task, algorithms need to modify the provided images according to the text instruction without affecting the unrelated components on the image. Typical methods for this task, e.g., StyleCLIP [33], rely on GAN inversion. Recently, diffusion models are adopted for this task for their superiority in reconstruction [19, 34], at the cost of re-training or fine-tuning existing diffusion models.

Here, we apply FlowGrad to this task. Given an image

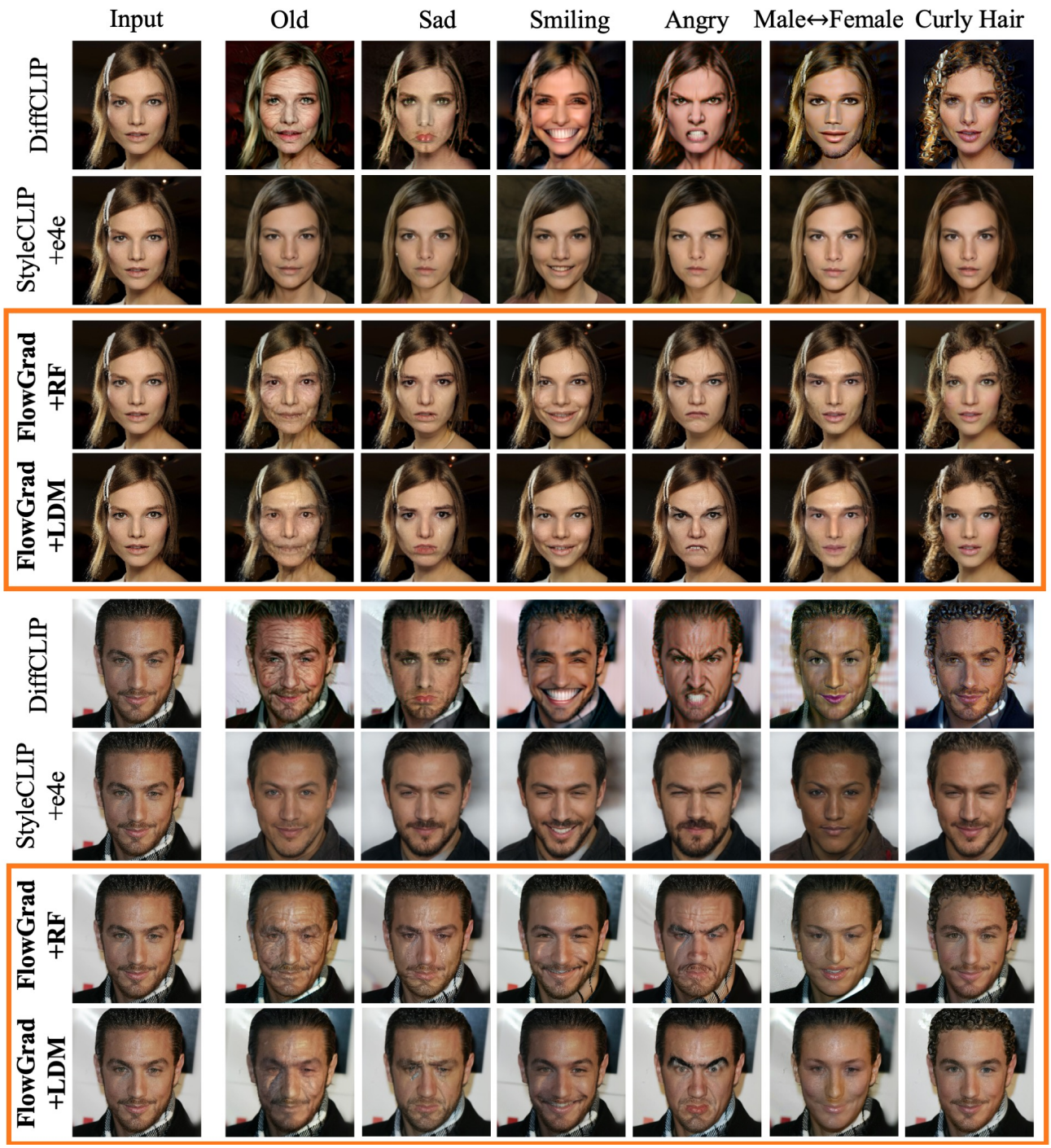


Figure 3. We provide qualitative comparison between different text-guided image manipulation algorithms, including DiffusionClip [19], StyleCLIP [33] with e4e encoder [41], FlowGrad + Rectified Flow (RF) [21] and FlowGrad + Latent Diffusion Model (LDM) [36]. The input images are sampled from CelebA-HQ dataset. The unconditional RF is pre-trained on CelebA-HQ and the unconditional LDM is pre-trained on FFHQ. We observe: (1) DiffusionCLIP causes unexpected changes in the style and color; (2) even with the state-of-the-art e4e encoder and the strong StyleGAN2 generator, StyleCLIP generates images with noticable change in identity and background; (3) ODEs can smoothly encode the input images and then edit the images by FlowGrad, without changing the unrelated elements. Note that FlowGrad uses *frozen* pre-trained ODEs.

Method	LPIPS (↓)	ID (↑)	CLIP (↑)
CG + RF [9]	0.346	0.643	0.292
CG + LDM [9]	0.383	0.513	0.298
DiffusionCLIP [19]	0.398	0.659	0.285
StyleCLIP [33]+e4e [41]	0.359	0.704	0.267
FlowGrad + RF	0.302	0.737	**0.299**
FlowGrad + LDM	**0.298**	**0.743**	0.294

Table 2. Quantitative comparison between different algorithms for editing images from CelebA-HQ. 'CG' refers to Classifier Guidance. We measure the LPIPS similarity and the identity similarity between the original image and the manipulated images to reflect faithfulness to the given image. Besides, we measure the Augmented CLIP score of the manipulated images to reflect closeness between the manipulated image and the text prompt. FlowGrad yields the highest faithfulness to both the provided image and the text prompt.

of interest x_g, our loss function is defined as,

$$\mathcal{L}(x_1) = \eta s(x_1, T) + (1-\eta)||x_1 - x_g||,$$

where $s(\cdot,\cdot)$ is the similarity score between image x_1 and text prompt T given by the CLIP model, $||x_1 - x_g||$ penalizes large variations from the original image x_g, and $\eta = 0.7$ for all the experiments. To avoid adversarial generation, we adopt the Augmented CLIP score [22] as $s(\cdot,\cdot)$. The hyperparameter λ is set to $1e-3$. We exploit the official pre-trained models including Latent Diffusion Model (LDM) [36] on FFHQ [17] and LSUN Church [46], and Rectified Flow (RF) [21] on CelebA-HQ [24]. We generate images with DDIM for LDM. For LDM, the Euler discretization step $N = 200$; for RF, $N = 100$. We set the step size $\alpha = 10.0$ and the number of optimization iterations $M = 10$. We compare with state-of-the-art text-guided image manipulation baselines, StyleCLIP [33] and DiffusionCLIP [19]. The details of the baseline configurations can be found in Appendix A. The qualitative results are shown in Figure 3, 4.

For quantitative comparison, we use the CelebA dataset, randomly sampled $1,000$ images, and manipulate them with text guidance. The text guidance contains {`old`, `sad`, `smiling`, `angry`, `curly hair`}. As in [19], we measure the face identity similarity, LPIPS similarity and augmented CLIP score, to respectively show the closeness to the original face, the original image and the text prompt. The results are shown in Table 2. Comparison of the running time between different methods is reported in Table 4.

5.2. Identify Global Semantic Direction in Pre-trained ODEs

Beyond editing individual images, we expect to find global semantic direction u that generalizes across for a wide range of images so that we can directly manipulate a new image without solving the optimization problem again. Global semantic directions can be found by optimizing the following loss function,

$$\mathcal{L} = \sum_{i=1}^{n} \eta s(x_1^{(i)}, T) + (1-\eta)||x_1^{(i)} - x_g^{(i)}||,$$

where $x_g^{(i)}$ are the training images. The global directions found in *frozen* pre-trained LDM-FFHQ and RF-CelebA are shown in Figure 6.

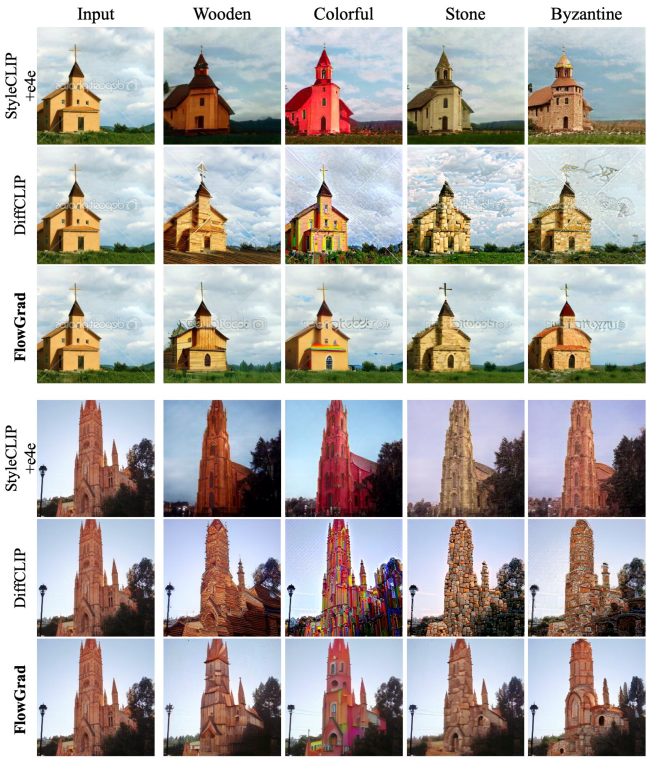
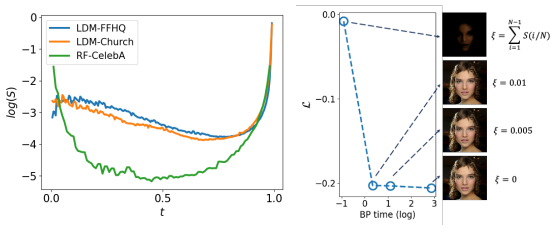


Figure 4. Qualitative comparison on LSUN Church. We apply FlowGrad on LDM-Church.



(a) (b)

Figure 5. (a)We show the average straightness for each time step k/N over 1,000 randomly sampled trajectories. For most of the time steps, $S(t)$ is below 0.001, implying an almost straight part. (b) We examine different choices of the threshold ξ. The x-axis is the running time of back-propagation in log scale. The y-axis is loss $\mathcal{L}$. Though one-step discretization ($\xi = \sum_{i=1}^{N-1} S(i/N)$) brings large approximation error and fails to generate meaningful results, $\xi \leq 0.01$ yields similar generation and loss.

Figure 6. We show global directions found in *frozen* pre-trained ODEs with FlowGrad. We randomly generate 10 images as the training set, and demonstrate the effectiveness of the obtained global directions on other synthesized images.

5.3. Ablation Study

In this section, we perform ablation studies to justify the design choices of FlowGrad. The ablations studies are conducted on text-guided image manipulation tasks.

Threshold ξ We examine the influence of the threshold ξ. We set $\xi = 0, 5e-3, 1e-2$, where $\xi = 0$ results in full back-propagation. Besides, we set $\xi = \sum_{i=1}^{N-1} S(i/N)$, which corresponds to the following one-step discretization,

$$x_1 = x_0 + v_\theta(x_0, 0).$$

We adopt RF-CelebA for this study. The results are shown in Figure 5b. We use the text prompt `curly hair`. We observe that one-step discretization brings large error, and $\xi < 1e-2$ results in images with similar visual quality and loss.

Sampling step N We examine the influence of reducing the number of sampling steps N. We use RF-CelebA. The results are shown in Figure 7. Smaller N generates over-smoothed images and fails the manipulation task.

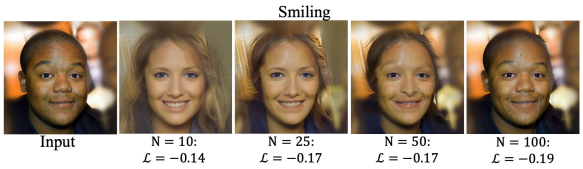


Figure 7. We investigate the influence of the number of Euler discretization steps N. When N is small, the discretization error becomes large and the generation quality degrades.

6. Conclusions

We propose FlowGrad, an efficient framework for controlled generation with ODE-based generative models using gradients. Our FlowGrad decomposes the back-propagation of the ODE trajecory, and compute the gradient with vector-Jacobian products. Moreover, by re-approximating the ODE trajectory with non-uniform discretization, FlowGrad can save 90% of the back-propagation time with small computation error. Experiments on text-guided image manipulation and global semantic direction detection shows the superiority of FlowGrad.

Acknowledgements

This research is supported by NSF CAREER1846421, SenSE2037267, EAGER-2041327, Office of Navy Research, and NSF AI Institute for Foundations of Machine Learning (IFML).

References

[1] Abdal, R., Qin, Y., Wonka, P.: Image2stylegan: How to embed images into the stylegan latent space? In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4432–4441 (2019) 2, 3, 5

[2] Abdal, R., Qin, Y., Wonka, P.: Image2stylegan++: How to edit the embedded images? In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8296–8305 (2020) 1, 2, 3, 5

[3] Bansal, A., Borgnia, E., Chu, H.M., Li, J.S., Kazemi, H., Huang, F., Goldblum, M., Geiping, J., Goldstein, T.: Cold diffusion: Inverting arbitrary image transforms without noise. arXiv preprint arXiv:2208.09392 (2022) 2

[4] Bao, F., Li, C., Zhu, J., Zhang, B.: Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. In: International Conference on Learning Representations 5

[5] Choi, J., Kim, S., Jeong, Y., Gwon, Y., Yoon, S.: Ilvr: Conditioning method for denoising diffusion probabilistic models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14367–14376 (2021) 5

[6] Chung, H., Sim, B., Ye, J.C.: Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12413–12422 (2022) 5

[7] Daras, G., Dean, J., Jalal, A., Dimakis, A.: Intermediate layer optimization for inverse problems using deep generative models. In: International Conference on Machine Learning. pp. 2421–2432. PMLR (2021) 1, 5

[8] Deng, J., Guo, J., Xue, N., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4690–4699 (2019) 11

[9] Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. Advances in Neural Information Processing Systems **34**, 8780–8794 (2021) 1, 5, 7, 12

[10] Grathwohl, W., Chen, R.T., Bettencourt, J., Sutskever, I., Duvenaud, D.: Ffjord: Free-form continuous dynamics for scalable reversible generative models. In: International Conference on Learning Representations (2018) 2

[11] Härkönen, E., Hertzmann, A., Lehtinen, J., Paris, S.: Ganspace: Discovering interpretable gan controls. Advances in Neural Information Processing Systems **33**, 9841–9850 (2020) 1, 5

[12] Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems **33**, 6840–6851 (2020) 1, 5

[13] Ho, J., Salimans, T.: Classifier-free diffusion guidance. In: NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications (2021) 1, 2

[14] Hoogeboom, E., Satorras, V.G., Vignac, C., Welling, M.: Equivariant diffusion for molecule generation in 3d. In: International Conference on Machine Learning. pp. 8867–8887. PMLR (2022) 12

[15] Huh, M., Zhang, R., Zhu, J.Y., Paris, S., Hertzmann, A.: Transforming and projecting images into class-conditional generative networks. In: European Conference on Computer Vision. pp. 17–34. Springer (2020) 5

[16] Karras, T., Aittala, M., Aila, T., Laine, S.: Elucidating the design space of diffusion-based generative models. arXiv preprint arXiv:2206.00364 (2022) 1, 5

[17] Karras, T., Laine, S., Aila, T.: 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) 7

[18] Kawar, B., Elad, M., Ermon, S., Song, J.: Denoising diffusion restoration models. In: ICLR Workshop on Deep Generative Models for Highly Structured Data (2022) 5

[19] Kim, G., Ye, J.C.: Diffusionclip: Text-guided image manipulation using diffusion models (2021) 1, 2, 5, 6, 7, 11, 12

[20] Kong, Z., Ping, W., Huang, J., Zhao, K., Catanzaro, B.: Diffwave: A versatile diffusion model for audio synthesis. In: International Conference on Learning Representations (2020) 5

[21] Liu, X., Gong, C., Liu, Q.: Flow straight and fast: Learning to generate and transfer data with rectified flow. arXiv preprint arXiv:2209.03003 (2022) 1, 2, 3, 5, 6, 7, 11

[22] Liu, X., Gong, C., Wu, L., Zhang, S., Su, H., Liu, Q.: Fusedream: Training-free text-to-image generation with improved clip+ gan space optimization. arXiv preprint arXiv:2112.01573 (2021) 2, 5, 7

[23] Liu, X., Wu, L., Ye, M., Liu, Q.: Let us build bridges: Understanding and extending diffusion generative models. arXiv preprint arXiv:2208.14699 (2022) 3, 5

[24] Liu, Z., Luo, P., Wang, X., Tang, X.: Large-scale celebfaces attributes (celeba) dataset. Retrieved August **15**(2018), 11 (2018) 7

[25] Lu, C., Zheng, K., Bao, F., Chen, J., Li, C., Zhu, J.: Maximum likelihood training for score-based diffusion odes by high order denoising score matching. In: International Conference on Machine Learning. pp. 14429–14460. PMLR (2022) 5

[26] Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., Zhu, J.: Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In: Advances in Neural Information Processing Systems 5

[27] Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., Van Gool, L.: Repaint: Inpainting using denoising diffusion probabilistic models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11461–11471 (2022) 5

[28] Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.Y., Ermon, S.: Sdedit: Guided image synthesis and editing with stochastic differential equations. In: International Conference on Learning Representations (2021) 1, 5

[29] Menon, S., Damian, A., Hu, S., Ravi, N., Rudin, C.: Pulse: Self-supervised photo upsampling via latent space exploration of generative models. In: Proceedings of the ieee/cvf

conference on computer vision and pattern recognition. pp. 2437–2445 (2020) 1, 5

[30] Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., Chen, M.: Glide: Towards photorealistic image generation and editing with text-guided diffusion models. arXiv preprint arXiv:2112.10741 (2021) 1, 5

[31] Pan, X., Zhan, X., Dai, B., Lin, D., Loy, C.C., Luo, P.: Exploiting deep generative prior for versatile image restoration and manipulation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021) 1, 5

[32] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019) 3, 4

[33] Patashnik, O., Wu, Z., Shechtman, E., Cohen-Or, D., Lischinski, D.: Styleclip: Text-driven manipulation of stylegan imagery. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2085–2094 (2021) 1, 5, 6, 7, 11, 12

[34] Preechakul, K., Chatthee, N., Wizadwongsa, S., Suwajanakorn, S.: Diffusion autoencoders: Toward a meaningful and decodable representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10619–10629 (2022) 5

[35] Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning. pp. 8748–8763. PMLR (2021) 1, 5

[36] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models (2021) 1, 6, 7

[37] Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S.K.S., Ayan, B.K., Mahdavi, S.S., Lopes, R.G., et al.: Photorealistic text-to-image diffusion models with deep language understanding. arXiv preprint arXiv:2205.11487 (2022) 5

[38] Shen, Y., Zhou, B.: Closed-form factorization of latent semantics in gans. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1532–1540 (2021) 1, 5

[39] Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: International Conference on Learning Representations (2020) 1, 2, 3, 5, 11

[40] Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. In: International Conference on Learning Representations (2020) 1, 2, 3, 5, 11

[41] Tov, O., Alaluf, Y., Nitzan, Y., Patashnik, O., Cohen-Or, D.: Designing an encoder for stylegan image manipulation. ACM Transactions on Graphics (TOG) **40**(4), 1–14 (2021) 3, 6, 7, 12

[42] Wu, L., Gong, C., Liu, X., Ye, M., Liu, Q.: Diffusion-based molecule generation with informative prior bridges. arXiv preprint arXiv:2209.00865 (2022) 1, 3, 5

[43] Xia, W., Zhang, Y., Yang, Y., Xue, J.H., Zhou, B., Yang, M.H.: Gan inversion: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022) 1, 5

[44] Xu, Y., Liu, Z., Tegmark, M., Jaakkola, T.: Poisson flow generative models. arXiv preprint arXiv:2209.11178 (2022) 1, 2, 5

[45] Ye, M., Wu, L., Liu, Q.: First hitting diffusion models for generating manifold, graph and categorical data. In: Advances in Neural Information Processing Systems (2022) 1

[46] Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., Xiao, J.: Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365 (2015) 7

[47] Zhao, M., Bao, F., Li, C., Zhu, J.: Egsde: Unpaired image-to-image translation via energy-guided stochastic differential equations. In: Advances in Neural Information Processing Systems 5

[48] Zhu, J., Shen, Y., Zhao, D., Zhou, B.: In-domain gan inversion for real image editing. In: European conference on computer vision. pp. 592–608. Springer (2020) 5

[49] Zhu, P., Abdal, R., Qin, Y., Femiani, J., Wonka, P.: Improved stylegan embedding: Where are the good latents? arXiv preprint arXiv:2012.09036 (2020) 1, 5