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.
2.1. Probability Flows

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

\[
dx = v_{\theta}(x, t) \, dt, \quad x_0 \sim \pi_0, \tag{1}
\]

Traditionally, \( v_{\theta} \) 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 \( v_{\theta} \) by regressing the target field \([3, 21, 44]\).

Once \( v_{\theta} \) is learned, one can simply sample from the ODE to get \( x_1 \) as drawn from the target distribution \( \pi_1 \) 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 \( \pi_0 \) 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.
Specifically, with \( v \) realized by solving the ODE forwardly, and backwardly, the trajectories are. In particular, if the trajectories accurately discretize and simulate the ODE depends on how the trajectory, then a single Euler step yields exact solution, \( x_1 = x_0 + v_\theta(x_0, 0) \).

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

\[
x_{k/N} = x_{(k+1)/N} - \frac{1}{N}v_\theta(x_{(k+1)/N}, (k+1)/N),
\]

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 \( L \) to control the end point \( x_1 \) towards our expectation by minimizing the following optimal control problem,

\[
\min_u L(x_1) + \lambda \int_0^1 ||u(t)||^2 dt,
\]

\[
s.t. \quad x_1 = x_0 + \int_0^1 (v_\theta(x_t, t) + u(t)) dt,
\]

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

\[
\min_u J := L(x_1) + \lambda \sum_{k=0}^{N-1} ||u_{k/N}||^2
\]

\[
s.t. \quad x_{(k+1)/N} = x_{k/N} + \frac{1}{N} \left( v_\theta(x_{k/N}, k/N) + u_{k/N} \right),
\]

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 \( \nabla u_{k/N} J \) needs to back-propagate through a chain with \( N - k \) nested \( v_\theta \). Since \( v_\theta \) 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,

\[
\nabla u_{k/N} J = (\nabla x_{(k+1)/N} J) \cdot J_{\phi_k}(x_{k/N}),
\]

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 \( \phi_k \) 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
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, \( \nabla_{z_1} J(x_1) \), is computed, we can get the gradient \( \nabla_{u_N} J = \frac{1}{N} \sum_{j=1}^{N} \nabla_{x_{j+1}} J \) by iteratively applying Eq. (8). Then we can then minimize \( J \) with gradient descent,

\[
    u_{k/N} \leftarrow u_{k/N} - \alpha \nabla_{u_{k/N}} J,
\]

where \( \alpha \) 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 \( \times \) 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,

\[
    x_{t_{j+1}} = x_{t_j} + (t_{j+1} - t_j)v_\theta(x_{t_j}, t_j),
\]

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:

**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,

\[
    S(k/N) = \max(d(v_{k-1}, v_k), d(u_{k+1}, v_k)),
\]

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,

\[
    t_0 = 0, \quad N(t_{j+1} - t_j) = \arg\min_m \left( \sum_{i=1}^{m} \frac{S(t_j + \frac{i}{N})}{S(t_j)} \geq \xi, \right), \quad (12)
\]

where \( 1 \leq m \leq N(1 - t_j) \) and stop when \( t_j = 1 \).

Here, \( \xi \) is a hyper-parameter to control the tolerance of straightness. As \( \xi \) increases, the minimal \( m \) becomes larger and the number of elements in \( G \) decreases. Thus, a large \( \xi \) 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,

\[
    \nabla_{x_{t_j}} J = (\nabla_{x_{t_{j+1}}} J) \cdot \phi'_j(x_{t_j}),
\]

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

**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.
learned from scratch [16, 21, 44], can map a latent code to noises retrieved from pre-trained diffusion models [39, 40] or directly handled 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 deterministic 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 problems 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.

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 gain 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 deterministic 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.

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 [3]. 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 problems 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.

5.1. 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 noticeable 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.
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.

<table>
<thead>
<tr>
<th>Method</th>
<th>LPIPS (↓)</th>
<th>ID (↑)</th>
<th>CLIP (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG + RF [9]</td>
<td>0.346</td>
<td>0.643</td>
<td>0.292</td>
</tr>
<tr>
<td>CG + LDM [9]</td>
<td>0.383</td>
<td>0.513</td>
<td>0.298</td>
</tr>
<tr>
<td>DiffusionCLIP [19]</td>
<td>0.398</td>
<td>0.659</td>
<td>0.285</td>
</tr>
<tr>
<td>StyleCLIP [33]+e4e [41]</td>
<td>0.359</td>
<td>0.704</td>
<td>0.267</td>
</tr>
<tr>
<td>FlowGrad + RF</td>
<td>0.302</td>
<td>0.737</td>
<td>0.299</td>
</tr>
<tr>
<td>FlowGrad + LDM</td>
<td>0.298</td>
<td>0.743</td>
<td>0.294</td>
</tr>
</tbody>
</table>

5.2. Identify Global Semantic Direction in Pre-trained ODEs

Beyond editing individual images, we expect to find global semantic direction \( \mathbf{u} \) 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 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 $\xi$ We examine the influence of the threshold $\xi$. 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.

6. Conclusions

We propose FlowGrad, an efficient framework for controlled generation with ODE-based generative models using

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.

gradients. Our FlowGrad decomposes the back-propagation of the ODE trajectory, 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).


