Generative Diffusion Models (Part 27): Step Size as Conditional Input

Jianlin Su

December 15, 2024

In this article, we once again focus on the sampling acceleration of diffusion models. As is well known, there are two main approaches to accelerating diffusion model sampling: one is to develop more efficient solvers, and the other is post-hoc distillation. However, according to the author's observation, except for SiD introduced in the previous two articles, these two schemes have rarely achieved results that reduce the number of generation steps to a single step. Although SiD can achieve single-step generation, it requires additional distillation costs and employs a GAN-like alternating training process during distillation, which always feels somewhat lacking.

The work to be introduced in this article is "One Step Diffusion via Shortcut Models". Its breakthrough idea is to treat the generation step size as a conditional input to the diffusion model and then add an intuitive regularization term to the training objective. This allows for the direct and stable training of a model capable of single-step generation, making it a simple yet effective classic work.

1 ODE Diffusion

The conclusions of the original paper are based on ODE-style diffusion models. Regarding the theoretical foundation of ODE-style diffusion, we have introduced it multiple times in parts (6), (12), (14), (15), and (17) of this series. One of the simplest ways to understand it is from the ReFlow perspective in (17), which we will briefly repeat here.

Assume $\boldsymbol{x}_0 \sim p_0(\boldsymbol{x}_0)$ is a random noise sampled from a prior distribution, and $\boldsymbol{x}_1 \sim p_1(\boldsymbol{x}_1)$ is a real sample sampled from the target distribution (Note: In previous articles, $\boldsymbol{x}_T$ was usually the noise and $\boldsymbol{x}_0$ was the target sample; here, they are reversed for convenience). ReFlow allows us to specify any motion trajectory from $\boldsymbol{x}_0$ to $\boldsymbol{x}_1$. The simplest trajectory is naturally a straight line:

$$\boldsymbol{x}_t = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1 \tag{1}$$

By taking the derivative on both sides, we can obtain the ODE (Ordinary Differential Equation) it satisfies:

$$\frac{d\boldsymbol{x}_t}{dt} = \boldsymbol{x}_1 - \boldsymbol{x}_0 \tag{2}$$

This ODE is simple, but in practice, it is useless because we want to generate $\boldsymbol{x}_1$ from $\boldsymbol{x}_0$ via the ODE, yet the above ODE explicitly depends on $\boldsymbol{x}_1$. To solve this problem, a simple idea is to "learn a function of $\boldsymbol{x}_t$ to approximate $\boldsymbol{x}_1 - \boldsymbol{x}_0$." After learning, we use it to replace $\boldsymbol{x}_1 - \boldsymbol{x}_0$, i.e.,

$$\boldsymbol{\theta}^* = \mathop{\text{argmin}}_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}_0\sim p_0(\boldsymbol{x}_0),\boldsymbol{x}_1\sim p_1(\boldsymbol{x}_1)}\left[\Vert\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - (\boldsymbol{x}_1 - \boldsymbol{x}_0)\Vert^2\right] \tag{3}$$

and

$$\frac{d\boldsymbol{x}_t}{dt} = \boldsymbol{x}_1 - \boldsymbol{x}_0 \quad\Rightarrow\quad \frac{d\boldsymbol{x}_t}{dt} = \boldsymbol{v}_{\boldsymbol{\theta}^*}(\boldsymbol{x}_t, t) \tag{4}$$

This is ReFlow. Of course, there is a missing theoretical proof here—that the ODE obtained by fitting $\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ through squared error indeed generates the desired distribution. For this part, readers can refer to "Generative Diffusion Models (17): General Steps for Constructing ODEs (Part 2)".

2 Step Size Self-consistency

Suppose we already have $\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$. Then, by solving the differential equation $\frac{d\boldsymbol{x}_t}{dt} = \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$, we can achieve the transformation from $\boldsymbol{x}_0$ to $\boldsymbol{x}_1$. The key point is "differential equation," but in reality, we cannot truly compute a differential equation numerically; we can only compute a "difference equation":

$$\boldsymbol{x}_{t+\epsilon} - \boldsymbol{x}_t = \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\epsilon \tag{5}$$

This difference equation is an "Euler approximation" of the original ODE. The degree of approximation depends on the step size ϵ. As $\epsilon \to 0$, it becomes exactly equal to the original ODE; in other words, the smaller the step size, the more accurate it is. However, the number of generation steps equals $1/\epsilon$, and we want the number of generation steps to be as small as possible. This means we cannot use a step size that is too small; ideally, ϵ could be equal to 1, so that $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_0, 0)$, completing the generation in one step.

The problem is that if we directly substitute a large step size into the above equation, the resulting $\boldsymbol{x}_1$ will inevitably deviate significantly from the exact solution. This is where the clever idea of the original paper (hereafter referred to as the "Shortcut Model") comes in: it posits that the model $\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ should not only be a function of $\boldsymbol{x}_t$ and t, but also a function of the step size ϵ. In this way, the difference equation (5) can adapt itself to the step size:

$$\boldsymbol{x}_{t+\epsilon} - \boldsymbol{x}_t = \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, \epsilon)\epsilon \tag{6}$$

The objective (3) trains the exact ODE model, so it trains the model for $\epsilon = 0$:

$$\mathcal{L}_1 = \mathbb{E}_{\boldsymbol{x}_0\sim p_0(\boldsymbol{x}_0),\boldsymbol{x}_1\sim p_1(\boldsymbol{x}_1)}\left[\frac{1}{2}\Vert\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, 0) - (\boldsymbol{x}_1 - \boldsymbol{x}_0)\Vert^2\right] \tag{7}$$

How is the part for $\epsilon > 0$ trained? Our goal is to have as few generation steps as possible, which is equivalent to saying we hope that "<u>taking 1 step with double the step size</u> equals <u>taking 2 steps with a single step size</u>":

$$\boldsymbol{x}_t + \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, 2\epsilon)2\epsilon = \underbrace{\boldsymbol{x}_t + \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, \epsilon)\epsilon}_{\tilde{\boldsymbol{x}}_{t+\epsilon}} + \boldsymbol{v}_{\boldsymbol{\theta}}(\underbrace{\boldsymbol{x}_t + \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, \epsilon)\epsilon}_{\tilde{\boldsymbol{x}}_{t+\epsilon}}, t+\epsilon, \epsilon)\epsilon \tag{8}$$

That is, $\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, 2\epsilon) = [\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, \epsilon) + \boldsymbol{v}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{t+\epsilon}, t+\epsilon, \epsilon)]/2$. To achieve this goal, we supplement a self-consistency loss function:

$$\mathcal{L}_2 = \mathbb{E}_{\boldsymbol{x}_0\sim p_0(\boldsymbol{x}_0),\boldsymbol{x}_1\sim p_1(\boldsymbol{x}_1)}\left[\Vert\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, 2\epsilon) - [\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, \epsilon) + \boldsymbol{v}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{t+\epsilon}, t+\epsilon, \epsilon)]/2\Vert^2\right] \tag{9}$$

The sum of $\mathcal{L}_1$ and $\mathcal{L}_2$ constitutes the loss function of the Shortcut Model.

(Note: Some readers have pointed out that the earlier "Consistency Trajectory Models: Learning Probability Flow ODE Trajectory of Diffusion" proposed using the start and end points of discretized time as conditional inputs. Once the start and end points are specified, the step size is actually determined, so Shortcut's approach of using step size as input is not entirely innovative.)

3 Model Details

The above covers almost all the theoretical content of the Shortcut Model, which is very elegant and concise. However, moving from theory to experiment requires some details, such as how the step size ϵ is integrated into the model.

First, when training $\mathcal{L}_2$, Shortcut does not sample ϵ uniformly from $[0, 1]$. Instead, it sets a minimum step size of 2^{-7} and then doubles them up to 1. Thus, all non-zero step sizes are limited to 8 values: $\{2^{-7}, 2^{-6}, 2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 1\}$. $\mathcal{L}_2$ is trained by uniformly sampling from the first 7 values. Consequently, there are only 9 possible values for ϵ (including 0). The Shortcut model directly inputs ϵ as an Embedding, which is added to the Embedding of t.

Secondly, note that the computational cost of $\mathcal{L}_2$ is higher than that of $\mathcal{L}_1$ because the term $\boldsymbol{v}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{t+\epsilon}, t+\epsilon, \epsilon)$ requires two forward passes. Therefore, the paper's approach is to use 3/4 of the samples in each batch to calculate $\mathcal{L}_1$ and the remaining 1/4 for $\mathcal{L}_2$. This operation not only saves computation but also adjusts the weights of $\mathcal{L}_1$ and $\mathcal{L}_2$. Since $\mathcal{L}_2$ is easier to train than $\mathcal{L}_1$, it can afford to have fewer training samples.

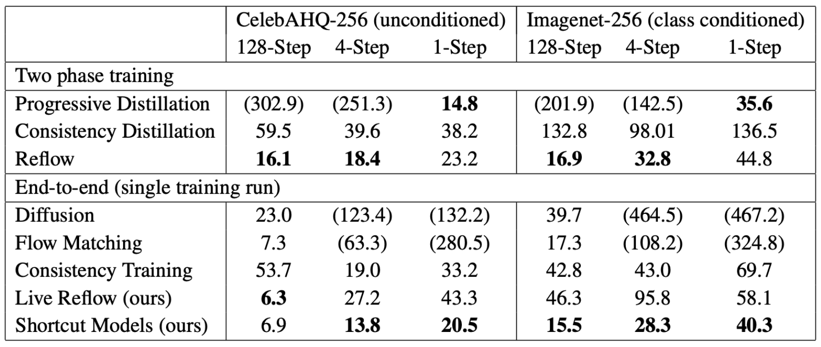
Additionally, in practice, the paper fine-tunes $\mathcal{L}_2$ by adding a stop gradient operator:

$$\mathcal{L}_2 = \mathbb{E}_{\boldsymbol{x}_0\sim p_0(\boldsymbol{x}_0),\boldsymbol{x}_1\sim p_1(\boldsymbol{x}_1)}\left[\|\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, 2\epsilon) - \mathrm{sg}[\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, \epsilon) + \boldsymbol{v}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{t+\epsilon}, t+\epsilon, \epsilon)]/2\|^2\right] \tag{10}$$

Why do this? According to the author's reply, this is a common practice in self-guided learning. The part under the stop gradient belongs to the target and should not have gradients, similar to unsupervised learning schemes like BYOL and SimSiam. However, in the author's view, the greatest value of this operation is saving training costs, as the term $\boldsymbol{v}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{t+\epsilon}, t+\epsilon, \epsilon)$ involves two forward passes; if backpropagation were required through it, the computation would double.

4 Experimental Results

Now let's look at the experimental results of the Shortcut Model. It appears to be the best-performing single-stage trained diffusion model for single-step generation currently available:



	CelebAHQ-256 (unconditioned) 128-Step	4-Step	1-Step	Imagenet-256 (class conditioned) 128-Step	4-Step	1-Step
Two phase training						
Progressive Distillation	(302.9)	(251.3)	**14.8**	(201.9)	(142.5)	**35.6**
Consistency Distillation	59.5	39.6	38.2	132.8	98.01	136.5
Reflow	**16.1**	**18.4**	23.2	**16.9**	**32.8**	44.8
End-to-end (single training run)						
Diffusion	23.0	(123.4)	(132.2)	39.7	(464.5)	(467.2)
Flow Matching	7.3	(63.3)	(280.5)	17.3	(108.2)	(324.8)
Consistency Training	53.7	19.0	33.2	42.8	43.0	69.7
Live Reflow (ours)	**6.3**	27.2	43.3	46.3	95.8	58.1
Shortcut Models (ours)	6.9	**13.8**	**20.5**	**15.5**	**28.3**	**40.3**

Table 1: **Comparison of training objectives under equivalent architecture (DiT-B) and compute.** FID-50k scores (lower is better) are shown over 128, 4, and 1-step denoising. Shortcut models provide high-quality samples under any inference budget, within a single training run. Compared to diffusion and flow-matching, shortcut models drastically reduce needed sampling steps. Compared to distillation approaches, short models simplify the training pipeline and provides flexibility to choose inference budget after training. Parentheses represent evaluation under conditions that the objective is not intended to support.

Figure 1: Quality evaluation of various diffusion models

Here is the actual sampling effect:

However, a close observation of the single-step generated samples reveals noticeable flaws. While the Shortcut Model has made significant progress compared to previous single-stage training schemes, there is still clear room for improvement.

The authors have open-sourced the code for the Shortcut Model. The GitHub link is:

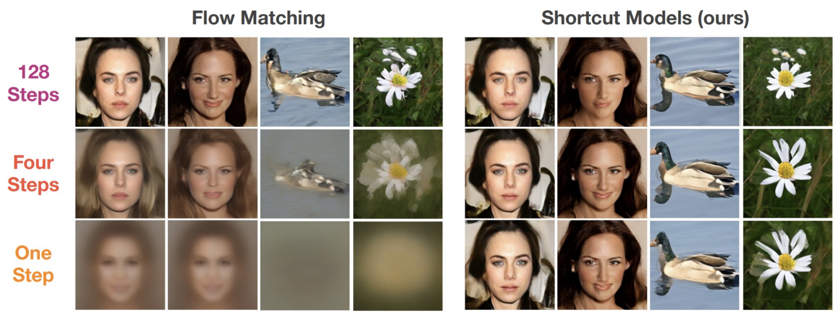


Figure 1: **Generations of flow-matching models and shortcut models for different inference budgets.** Shortcut models generate high-quality images across a wide range of inference budgets, including using a single forward pass, drastically reducing sampling time by up to 128x compared to diffusion and flow-matching models. In contrast, diffusion and flow-matching models rapidly deteriorate when queried in the few-step setting. The same starting noise used within each column and two models are trained on CelebA-HQ and Imagenet-256 (class conditioned).

Figure 2: Comparison of actual sampling effects between Flow Matching and Shortcut Model

https://github.com/kvfrans/shortcut-models

Incidentally, the Shortcut Model was submitted to ICLR 2025 and received unanimous praise from reviewers (all scores of 8).

5 Extended Thinking

Seeing the Shortcut Model, what related works come to mind? One that might be unexpected is AMED, which we introduced in "Generative Diffusion Models (Part 21): Accelerating ODE Sampling with the Mean Value Theorem".

The underlying ideas of the Shortcut Model and AMED are connected. Both recognize that relying solely on complex high-order solvers to reduce the NFE (Number of Function Evaluations) to single digits is already difficult, let alone achieving single-step generation. Thus, they agree that what needs to change is not the solver, but the model itself. How should it change? AMED thought of the "Mean Value Theorem": by integrating both sides of the ODE, we have the exact:

$$\boldsymbol{x}_{t+\epsilon} - \boldsymbol{x}_t = \int_t^{t+\epsilon} \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_\tau, \tau) d\tau \tag{11}$$

By analogy with the Mean Value Theorem for Integrals, we can find an $s \in [t, t+\epsilon]$ such that:

$$\frac{1}{\epsilon}\int_t^{t+\epsilon} \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_\tau, \tau) d\tau = \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_s, s) \tag{12}$$

Thus we get:

$$\boldsymbol{x}_{t+\epsilon} - \boldsymbol{x}_t = \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_s, s)\epsilon \tag{13}$$

Of course, the Mean Value Theorem for Integrals strictly applies only to scalar functions, not necessarily to vector functions, hence the "analogy." The problem is that the value of s is unknown, so AMED's approach is to use a very small model (with negligible computation) to predict s.

AMED is a post-hoc correction method based on existing diffusion models, so its effectiveness depends on how well the Mean Value Theorem holds for the $\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ model, which involves some "luck." Furthermore, AMED needs to use an Euler step to estimate $\boldsymbol{x}_s$ first, so its minimum NFE is 2, and it cannot achieve single-step generation. In contrast, the Shortcut Model is more "aggressive"; it directly treats the step size as a conditional input and uses the acceleration condition (8) as a loss function. This not only avoids the feasibility discussion of the "Mean Value Theorem" approximation but also allows the minimum NFE to be reduced to 1.

More cleverly, upon closer reflection, we find commonalities in their approaches. As mentioned, Shortcut directly converts ϵ into an Embedding and adds it to the Embedding of t. Isn't this equivalent to modifying t, just as AMED does? The difference is that AMED directly modifies the numerical value of t, while Shortcut modifies the Embedding of t.

6 Summary

This article introduced a new work on diffusion models that achieves single-step generation through single-stage training. Its breakthrough idea is to treat the step size as a conditional input to the diffusion model, paired with an intuitive regularization term. This allows a single-step generation model to be obtained through a single stage of training.