SoFlow: Solution Flow Models for One-Step Generative Modeling

Anonymous authors
Paper under double-blind review

Abstract

The multi-step denoising process in diffusion and Flow Matching models causes major efficiency issues, which motivates research on few-step generation. We present Solution Flow Models (SoFlow), a framework for one-step generation from scratch. By analyzing the relationship between the velocity function and the solution function of the velocity Ordinary Differential Equation (ODE), we propose a flow matching loss and a solution consistency loss to train our models. The flow matching loss allows our models to provide estimated velocity fields for Classifier-Free Guidance (CFG) during training, which improves generation performance. Notably, our consistency loss does not require the calculation of the Jacobian-Vector Product (JVP), a common requirement in recent works that is not well-optimized in deep learning frameworks like PyTorch. Experimental results indicate that, when trained from scratch using the same diffusion transformer (DiT) architecture and with an equal number of training epochs, our models achieve better FID-50K scores compared to MeanFlow models on the ImageNet 256x256 dataset.

Figure 1: Generated one-step samples on Imagenet 256×256 dataset by our Solution Flow Models.

1 Introduction

Diffusion models (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Ho et al., 2020; Song et al., 2020) and Flow Matching models (Lipman et al., 2022; Liu et al., 2022; Albergo & Vanden-Eijnden, 2022) have emerged as foundational frameworks in generative modeling. Diffusion models operate by systematically adding noise to data and then learning a denoising process to generate high-quality samples. Flow Matching offers a more direct alternative, modeling the velocity fields that transport a simple prior distribution to a complex data distribution. Despite their power and success across various generative tasks (Esser et al., 2024; Ma et al., 2024; Polyak et al., 2024), both approaches rely on an iterative, multi-step sampling process, which hinders their generation efficiency.

Addressing this latency is becoming a key area of research. Consistency Models (Song et al., 2023; Song & Dhariwal, 2023; Geng et al., 2024; Lu & Song, 2024) and related techniques (Kim et al., 2023; Wang et al., 2024a; Frans et al., 2024; Heek et al., 2024) have gained prominence by enabling rapid, few-step generation. These methods learn a direct mapping from any point on a generative trajectory to a consistent, "clean" output, bypassing the need for iterative refinement. However, this paradigm introduces significant challenges. Consistency models trained from scratch often fail to leverage Classifier-Free Guidance (CFG) for enhancing sample quality, and they are further hindered by instability caused by changing optimization targets. Recent works (Peng et al., 2025; Geng et al., 2025) address this instability by incorporating a flow matching loss. However, this approach introduces a new computational bottleneck: it relies on Jacobian-Vector Product (JVP) calculations, which are much less optimized than forward propagation in frameworks such as PyTorch.

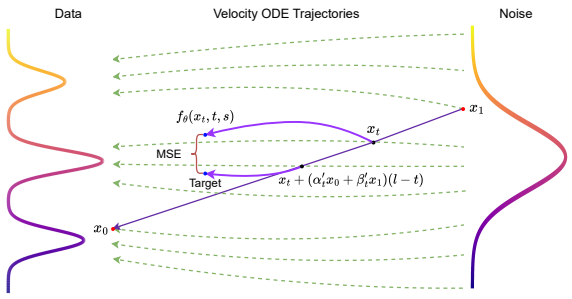

Figure 2: **Illustration of solution consistency loss:** The plot shows a straight-line flow matching trajectory defined by a data-noise pair (x_0, x_1). Given three time points $s < l < t$, two positions x_t and $x_t + (\alpha'_t x_0 + \beta'_t x_1)(l-t)$ are determined on the trajectory between x_0 and x_1 to compute the mean square error between our model $f_\theta(x_t, t, s)$ and the stop-gradient target term $f_{\theta^-}(x_t + (\alpha'_t x_0 + \beta'_t x_1)(l-t), l, s)$.

In this work, we introduce a new approach for one-step generation that avoids these limitations. Instead of relying on iterative ODE solvers, we propose to directly learn the solution function of the velocity ODE defined by Flow Matching. We denote this function as $f(x_t, t, s)$, which explicitly maps a state x_t at time t to its evolved state x_s at any other time s. To learn it with a parameterized model $f_\theta(x_t, t, s)$, we first analyze the properties that can make a neural network into a valid solution function. Based on this analysis, we formulate a training objective comprising a flow matching loss and a solution consistency loss. The resulting model, $f_\theta(x_t, t, s)$, not only accommodates the flow matching objective and CFG naturally but also eliminates the need for expensive JVP calculations during training. Our experimental results demonstrate the effectiveness of this approach, showing that our models achieve superior FID-50K scores compared to MeanFlow models on the ImageNet 256×256 dataset, using the same diffusion transformer (DiT) architecture and training duration.

2 Related Works

Diffusion, Flow Matching and Stochastic Interpolants Diffusion models (Sohl-Dickstein et al., 2015; Song et al., 2020; Kingma et al., 2021; Karras et al., 2022) and Flow Matching (Lipman et al., 2022; Liu et al., 2022) are widely adopted generative modeling frameworks. These approaches either progressively corrupt data with noise and train a neural network to denoise it, or learn a velocity field that governs a transformation from the data distribution to a simple prior. They have been scaled successfully for image generation (Rombach et al., 2022; Saharia et al., 2022; Podell et al., 2023) and video generation (Ho et al., 2022; Brooks et al., 2024) tasks. Stochastic interpolants (Albergo & Vanden-Eijnden, 2022; Albergo et al., 2023) build upon these concepts by explicitly defining stochastic trajectories between the data and prior distributions and aligning their associated velocity fields to enable effective distributional transport.

Few-step Generative Models Reducing the number of sampling steps has become an active research direction, driven by the need for faster generation and better theoretical insights. Prior efforts fall into two main streams: (i) distillation-based approaches that compress pre-trained multi-step models into few-step generators (Salimans & Ho, 2022; Sauer et al., 2024; Geng et al., 2023; Luo et al., 2023b; Yin et al., 2024; Zhou et al., 2024); and (ii) from-scratch training, which learns fast samplers without teachers, most prominently through the consistency-training family (CMs, iCT, ECT, sCT) (Song et al., 2023; Song & Dhariwal, 2023; Geng et al., 2024; Lu & Song, 2024).

Consistency Models (CMs) collapse noised inputs directly to clean data, enabling one- or few-step generation (Song et al., 2023). While first applied mainly in distillation (Luo et al., 2023a; Geng et al., 2024), later works established that they can also be trained from scratch via consistency training (Song & Dhariwal, 2023). Building on this, iCT (Song & Dhariwal, 2023) simplifies objectives and improves stability with robust losses and teacher-free training, while ECT (Geng et al., 2024) introduces a continuous-time ODE formulation that unifies CMs and diffusion. The recent sCT framework (Lu & Song, 2024) further stabilizes continuous-time consistency training, scaling CMs up to billion-parameter regimes.

Beyond fixed start/end settings, newer approaches extend CMs to arbitrary timestep transitions, aligning better with continuous-time dynamics. Shortcut models (Frans et al., 2024) condition on noise level and step size to flexibly support both one- and few-step sampling with shared weights. MeanFlow (Geng et al., 2025) introduces interval-averaged velocities and analyzes the relationship

between averaged and instantaneous velocities. IMM (Zhou et al., 2025) matches moments across transitions in a single-stage objective, reducing variance and avoiding multi-stage distillation. Flow-Anchored CMs (Peng et al., 2025) regularize shortcut learning with a flow matching anchor, improving stability and generalization. On the distillation side, Align Your Flow (Sabour et al., 2025) unifies CM and FM into continuous-time flow maps effective across arbitrary step counts, further enhanced with autoguidance and adversarial fine-tuning. Finally, Transition Models (Wang et al., 2025) reformulate generation around exact finite-interval dynamics, unifying few- and many-step regimes and achieving monotonic quality gains as step budgets increase.

3 Preliminary: Flow Matching

We first introduce the setting of Flow Matching. Flow Matching models learn a velocity field that transforms a known prior distribution like standard Gaussian distribution to the data distribution. More precisely, we denote the data distribution as $p(x_0)$, and the prior distribution as $p(x_1)$, which is standard Gaussian distribution $N(0, I)$ in our setting.

A general noising process is defined as $x_t = \alpha_t x_0 + \beta_t x_1$, where $x_t \in \mathbb{R}^n, t \in [0, 1]$, α_t and β_t are continuously differentiable functions satisfying the boundary conditions $\alpha_0 = 1$, $\alpha_1 = 0$, $\beta_0 = 0$, and $\beta_1 = 1$. Then, the marginal velocity field associated with the noising process is defined as

$$v(x_t, t) = \mathbb{E}_{p(x_0, x_1|x_t)}[\alpha'_t x_0 + \beta'_t x_1], \tag{1}$$

which is a conditional expectation of the conditional velocity field. According to the Flow Matching framework, given the marginal velocity field $v(x_t, t)$, new samples can be generated by first sampling $x_1 \sim p(x_1)$, and then solving the initial value problem (IVP) of the following velocity ordinary differential equation (ODE) from $t = 1$ to $t = 0$:

$$\frac{\mathrm{d}X(t)}{\mathrm{d}t} \;=\; v(X(t), t), X(1) = x_1. \tag{2}$$

To construct a generative model based on this principle, a straightforward approach is to approximate the marginal velocity field using a neural network v_θ, parameterized by θ, which is trained by minimizing the following mean square objective:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t, x_t} \left\| v_\theta(x_t, t) - v(x_t, t) \right\|^2. \tag{3}$$

However, directly optimizing a neural network with this loss is impractical since the real velocity field $v(x, t)$ is a conditional expectation given x_t. To overcome this challenge, a conditional variant of the flow matching loss is introduced (Lipman et al., 2022; Liu et al., 2022):

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t, x_0, x_1, x_t} \left\| v_\theta(x_t, t) - (\alpha'_t x_0 + \beta'_t x_1) \right\|^2. \tag{4}$$

Minimizing $\mathcal{L}_{\mathrm{CFM}}$ is equivalent to minimizing the original objective $\mathcal{L}_{\mathrm{FM}}$. This loss function is tractable since we only need to sample data-noise pairs to construct targets for the network.

4 Solution Flow Models

4.1 Formulations

The main purpose of this work is to investigate how can we train a neural network that can directly solve the velocity ODE, thereby eliminating the reliance on numerical ODE solvers in Flow Matching models. We study this problem under the assumption that the velocity field $v(x_t, t) \in \mathbb{R}^n$, with $x_t \in \mathbb{R}^n$ and $t \in [0, 1]$, is continuously differentiable and globally Lipschitz continuous with respect to x_t, i.e. $||v(x_t, t) - v(x'_t, t)||_2 \leq L||x_t - x'_t||_2, \forall x_t, x'_t \in \mathbb{R}^n, t \in [0, 1]$.

According to the existence and uniqueness theorem of ODEs, these two assumptions guarantee that given initial condition $X(t) = x_t$, where $x_t \in \mathbb{R}^n, t \in [0, 1]$ can be arbitrarily chosen, a unique solution $X(s), s \in [0, t]$ to the velocity ODE $\frac{dX(s)}{ds} = v(X(s), s)$ exists. Since the initial condition can be varied, we denote this unique solution by notation $f(x_t, t, s)$. Then we immediately have two equations by ODE's definition, for any $x_t \in \mathbb{R}^n, 0 \leq s \leq t \leq 1$:

$$f(x_t, t, t) = x_t, \partial_3 f(x_t, t, s) = v(f(x_t, t, s), s) \tag{5}$$

where $\partial_3 f(x_t, t, s) \in \mathbb{R}^n$ is the partial derivative function of $f(x_t, t, s)$ with respect to the third variable. Since $f(x_t, t, s)$ maps an initial value x_t at time t to the unique solution of the velocity ODE at time s, we denote it as the solution function in this paper. Owing to the continuous differentiability of the velocity field $v(x_t, t)$, the solution function $f(x_t, t, s)$ is also continuously differentiable. To realize one-step generation, we need to train a model $f_\theta(x_t, t, s)$ to approximate the ground truth solution function $f(x_t, t, s)$, which is determined uniquely by the velocity field $v(x_t, t)$.

Under the setting discussed above, the following two conditions are sufficient to ensure that $f_\theta(x_t, t, s) = f(x_t, t, s)$ for all $x_t \in \mathbb{R}^n, 0 \le s \le t \le 1$:

$$f_\theta(x_t, t, t) = x_t, \partial_1 f_\theta(x_t, t, s) v(x_t, t) + \partial_2 f_\theta(x_t, t, s) = 0. \tag{6}$$

where $\partial_1 f_\theta(x_t, t, s) \in \mathbb{R}^{n \times n}$ is the Jacobian matrix function of $f_\theta(x_t, t, s)$ with respect to the first variable, $v(x_t, t)$ is multiplied by it as a matrix-vector product, and $\partial_2 f_\theta(x_t, t, s) \in \mathbb{R}^n$ is the partial derivative vector function of $f_\theta(x_t, t, s)$ with respect to the second variable.

This can be proven by expanding the following derivative for $0 \le s \le l \le t \le 1$:

$$\frac{\mathrm{d}}{\mathrm{d}l}(f_\theta(f(x_t, t, l), l, s)) = \partial_1 f_\theta(f(x_t, t, l), l, s) \underbrace{\partial_3 f(x_t, t, l)}_{=v(f(x_t,t,l),l)} + \partial_2 f_\theta(f(x_t, t, l), l, s) = 0. \tag{7}$$

The expression is equal to 0 due to the first condition in Eq. 6. Thus, we know that the model $f_\theta(x_t, t, s)$ is indeed the true solution function $f(x_t, t, s)$ by:

$$f_\theta(x_t, t, s) = f_\theta(f(x_t, t, t), t, s) = f_\theta(f(x_t, t, s), s, s) = f(x_t, t, s), \tag{8}$$

where we use the property that $f(x_t, t, t) = f_\theta(x_t, t, t) = x_t$ and $f_\theta(f(x_t, t, l), l, s)$ is invariant with respect to l, allowing us to set $l = t$ and $l = s$ to finish the proof.

4.2 Learning Objectives

Now we consider how to build efficient objectives for neural networks to learn the ground truth solution operator, according to the two conditions mentioned in the previous section. To ensure the first boundary condition of Eq. 6 is satisfied, we adopt the following parameterization:

$$f_\theta(x_t, t, s) = a(t, s)x_t + b(t, s)F_\theta(x_t, t, s),$$

where $a(t, s)$ and $b(t, s)$ are continuously differentiable scalar functions satisfying $a(t, t) = 1$ and $b(t, t) = 0$ for all $t \in [0, 1]$, and $F_\theta(x, t, s)$ denotes a raw neural network. Following the notations in the last section, we use $\partial_1 a(t, s), \partial_2 a(t, s), \partial_1 b(t, s), \partial_2 b(t, s)$ to denote the partial derivatives of $a(t, s)$ and $b(t, s)$ with respect to the first and second variables. In our experiments, we choose two specific parameterizations, including the Euler parameterization and the triangular parameterization:

$$f_\theta(x_t, t, s) = x_t + (s - t)F_\theta(x_t, t, s),$$

$$f_\theta(x_t, t, s) = \cos\frac{\pi}{2}(s - t)x_t + \sin\frac{\pi}{2}(s - t)F_\theta(x_t, t, s).$$

Obviously, these two parameterizations satisfy the boundary condition $f_\theta(x_t, t, t) = x_t$. Then, we construct two loss functions for training using Eq. 6.

Flow Matching Loss Firstly, we consider a special situation of Eq. 6 when $t = s$. Recall that the boundary condition of Eq. 6 gives $f_\theta(x_t, t, t) = x_t, \forall x_t \in \mathbb{R}^n, t \in [0, 1]$, we have:

$$\partial_1 f_\theta(x_t, t, t) = I_n, 0 = \left.\frac{\mathrm{d} f_\theta(x_t, l, l)}{\mathrm{d}l}\right|_{l=t} = \partial_2 f_\theta(x_t, t, t) + \partial_3 f_\theta(x_t, t, t) \tag{9}$$

where $I_n \in \mathbb{R}^{n \times n}$ is the identity matrix. Then Eq. 6 can be simplified to $v(x_t, t) = \partial_3 f_\theta(x_t, t, t)$. Under our parameterization mentioned above, we have

$$v(x_t, t) = \partial_3 f_\theta(x_t, t, t) = \partial_2 a(t, t)x_t + \partial_2 b(t, t)F_\theta(x_t, t, t) + \underbrace{b(t, t)}_{0} \partial_3 F_\theta(x_t, t, t). \tag{10}$$

where the complex term containing $\partial_3 F_\theta(x_t, t, t)$ is canceled since $b(t, t) = 0$ by our choice of parameterization. Now we obtain a flow matching loss for our neural networks:

$$L_{\mathrm{FM}}(\theta) = \mathbb{E}_{t,x_0,x_1,x_t}\left[\frac{w_{\mathrm{FM}}(t, \mathrm{MSE})}{D}\left\|\partial_2 a(t,t)x_t + \partial_2 b(t,t)F_\theta(x_t,t,t) - (\alpha_t' x_0 + \beta_t' x_1)\right\|_2^2\right], \tag{11}$$

where D is the data dimension, $\alpha_t' x_0 + \beta_t' x_1$ is used to replace the intractable marginal velocity field $v(x_t, t) = \mathbb{E}_{p(x_0,x_1|x_t)}[\alpha_t' x_0 + \beta_t' x_1]$ during the training process following the standard Flow Matching framework. In addition, this velocity term can also be provided by a teacher model in distillation situations, but in this paper, we focus on the situation of training from scratch.

Previous works (Geng et al., 2024; 2025) have demonstrated that choosing an adaptive weighting function is beneficial for few-step generative models. Following their approach, we choose

$$w_{\mathrm{FM}}(t, \mathrm{MSE}) = \frac{1}{|\partial_2 b(t,t)|(\mathrm{MSE} + \epsilon)^p} \tag{12}$$

as our weighting function, where $|\partial_2 b(t, t)|$ is used to balance the raw network's gradients across time, and MSE represents the original mean squared error. Here ϵ is a smoothing factor to prevent excessively small values, and p is a factor that determines how robust the loss is. For $p = 0$, the objective degenerates to the mean squared error. For $p > 0$, this factor will penalize the data points with large errors in a data batch to make the objective more robust.

The original Flow Matching framework samples t uniformly, while more recent works (Esser et al., 2024; Geng et al., 2025) suggest sampling t from a logit-normal distribution. We also sample t from $\sigma(\mathcal{N}(\mu_{\mathrm{FM}}, \sigma_{\mathrm{FM}}^2))$, where $\sigma(\cdot)$ and $\mathcal{N}$ represent the sigmoid function and normal distribution.

Solution Consistency Loss We now consider how to build a training target for the $s < t$ situation using Eq. 6. According to the Taylor's expansion, we have an approximation equation:

$$\frac{f_\theta(x_t, t, s) - f_\theta(x_t + v(x_t,t)(l-t), l, s)}{t - l} = (\partial_1 f_\theta(x_t, t, s)v(x_t, t) + \partial_2 f_\theta(x_t, t, s)) + o(1), \tag{13}$$

where $l \in (s, t)$ is close to t. Thus, we can adopt the following objective:

$$L_{\mathrm{SCM}}(\theta) = \mathop{\mathbb{E}}_{\substack{t,l,s,\\ x_0,x_1,x_t}}\left[\frac{w_{\mathrm{SCM}}(t,l,s,\mathrm{MSE})}{D}\left\|f_\theta(x_t,t,s) - f_{\theta^-}\big(x_t + (\alpha_t' x_0 + \beta_t' x_1)(l-t), l, s\big)\right\|_2^2\right], \tag{14}$$

where D is the data dimension, θ^- means taking the stop-gradient operation to the parameters, and $(\alpha_t' x_0 + \beta_t' x_1)$ is again used to replace the intractable $v(x_t, t) = \mathbb{E}_{p(x_0,x_1|x_t)}[\alpha_t' x_0 + \beta_t' x_1]$ following the common practice. The adaptive weighting is chosen as follows:

$$w_{\mathrm{SCM}}(t, l, s, \mathrm{MSE}) = \frac{1}{(t-l)|b(t,s)|} \times \frac{1}{(\frac{\mathrm{MSE}}{(t-l)^2} + \epsilon)^p}, \tag{15}$$

where the first term ensures the gradient magnitude of the raw network $F_\theta(x_t, t, s)$ is stable, and the second term is again used to provide a more robust loss by scaling down the coefficient for data points with large mean square errors when $p > 0$. Besides, we divide the mean square error in the adaptive term by $(t - l)^2$ since its magnitude is proportional to $(t - l)^2$.

As for the sampling method of t, l, s during training, we first sample t and s from two logit-normal distributions, $\sigma(\mathcal{N}(\mu_t, \sigma_t^2))$ and $\sigma(\mathcal{N}(\mu_s, \sigma_s^2))$, respectively. Here, s is clamped to ensure $s < t - 10^{-4}$. We then determine l using the following method:

$$l = t + (s - t) \times r(k, K), \tag{16}$$

where k, K represent the current and total training steps, respectively. The function $r(k, K)$ represents a monotonically decreasing schedule that gradually moves l towards t throughout training. To avoid numerical issues, l is clamped to ensure $l < t - 10^{-4}$. This schedule decreases from an initial value r_{init} to an end value r_{end}. In our ablation studies, we test exponential, cosine, linear, and constant schedules. For more implementation details, please refer to Appendix A.

The total training loss is a combination of the flow matching and solution consistency losses: $L(\theta) = \lambda L_{\mathrm{FM}}(\theta) + (1 - \lambda)L_{\mathrm{SCM}}(\theta)$. The parameter λ controls the balance between them by determining the fraction of a data batch dedicated to computing $L_{\mathrm{FM}}(\theta)$, while the remaining fraction is used for $L_{\mathrm{SCM}}(\theta)$. We perform ablation studies to determine the optimal value of λ.

4.3 CLASSIFIER-FREE GUIDANCE

Classifier-Free Guidance (CFG) (Ho & Salimans, 2022) is a standard technique in diffusion models for enhancing conditional generation. The models are trained with randomly dropped conditions to mix conditional and unconditional data. During inference, CFG is applied by linearly combining predictions from the label-conditional and unconditional models to enhance generation quality.

Unlike this approach, since our goal is to let our models directly solve the velocity ODE, we need to train them directly with the linearly combined guided marginal velocity field, rather than training them separately on conditional and unconditional marginal velocity fields and combining their solutions linearly. Mathematically, the label-conditional marginal velocity field is defined as:

$$v(x_t, t \mid c) = \mathbb{E}_{p(x_0, x_1 \mid x_t, c)}[\alpha'_t x_0 + \beta'_t x_1], \tag{17}$$

With classifier-free guidance, the guided marginal velocity field is given by:

$$v_g(x_t, t, c) = wv(x_t, t \mid c) + (1 - w)v(x_t, t), \tag{18}$$

Notably, this velocity field depends purely on the data distribution and is independent of the models. Our goal is for the model to directly solve the velocity ODE defined by this guided field. However, since the unconditional velocity field is not directly accessible in conditional generation settings, we must train the network to also predict the unconditional velocity field for CFG training.

In implementation, we randomly replace the label c with an empty label ϕ with probability 0.1 in both the flow matching loss and the solution consistency loss. For data points assigned the empty label ϕ, we compute the loss functions directly by passing ϕ to the network and evaluating according to Eq. 11 and Eq. 14. For data points with a non-empty label c, we first predict the unconditional velocity field v_{uncond} using $\partial_2 a(t,t)x_t + \partial_2 b(t,t)F_\theta(x_t, t, t, \phi)$ as shown in Eq. 10. We then compute the guided loss by passing c to the network and replacing the velocity term $\alpha'_t x_0 + \beta'_t x_1$ in Eq. 11 and Eq. 14 with $w(\alpha'_t x_0 + \beta'_t x_1) + (1 - w)v_{\text{uncond}}$, where w is the CFG strength.

It is worth mentioning that the term $w(\alpha'_t x_0 + \beta'_t x_1) + (1 - w)v_{\text{uncond}}$ typically exhibits larger variance compared to the original term $\alpha'_t x_0 + \beta'_t x_1$, due to the scaling effect of w. This increased variance adversely affects both training convergence speed and final performance. Note that the model also learns the guided velocity field through the flow matching loss when the label c is provided. Therefore, we can predict v_{guided} using $\partial_2 a(t,t)x_t + \partial_2 b(t,t)F_\theta(x_t, t, t, c)$, and then employ

$$v_{\text{mix}} = m(w(\alpha'_t x_0 + \beta'_t x_1) + (1 - w)v_{\text{uncond}}) + (1 - m)v_{\text{guided}} \tag{19}$$

to replace the original term $\alpha'_t x_0 + \beta'_t x_1$, where $0 < m \leq 1$ is the velocity mix ratio. Since the majority of the variance originates from the stochastic term $\alpha'_t x_0 + \beta'_t x_1$, using a small m can effectively reduce the variance (v_{guided} is a model prediction value with much lower variance.)

After introducing the flow matching and solution consistency losses and demonstrating how to apply CFG during training, we now present the pseudo-code for our model's guided training process to provide a more comprehensive understanding.

Algorithm 1 Train Solution Flow Models with CFG

Input: model $f_\theta(x_t, t, s, c)$, flow matching data ratio λ, CFG strength w, velocity mix ratio m
Sample a data batch: $x_0, c \sim p_{data}(x_0, c)$
Split data for two losses by λ: $(x_0^{\text{FM}}, x_0^{\text{SCM}}) = x_0, (c^{\text{FM}}, c^{\text{SCM}}) = c$
Sample $t_{\text{FM}}, t_{\text{SCM}}, l_{\text{SCM}}, s_{\text{SCM}}$ according to subsection 4.2
Get $x_t^{\text{FM}}, v_t^{\text{FM}}, x_t^{\text{SCM}}, v_t^{\text{SCM}}$ by standard Flow Matching framework
Compute $v_{\text{mix}}^{\text{FM}}$ and $v_{\text{mix}}^{\text{SCM}}$ by Eq. 19 with w and m
Randomly replace $v_{\text{mix}}^{\text{FM}}$ and $v_{\text{mix}}^{\text{SCM}}$ by v_t^{FM} and v_t^{SCM} with CFG drop rate 0.1
Compute $L_{\text{FM}}(\theta)$ according to Eq. 11 by replacing $(\alpha'_t x_0 + \beta'_t x_1)$ term with $v_{\text{mix}}^{\text{FM}}$
Compute $L_{\text{SCM}}(\theta)$ according to Eq. 14 by replacing $(\alpha'_t x_0 + \beta'_t x_1)$ term with $v_{\text{mix}}^{\text{SCM}}$
Update f_θ via gradient descent according to $L(\theta) = \lambda L_{\text{FM}}(\theta) + (1 - \lambda)L_{\text{SCM}}(\theta)$

Regarding the inference process of our model, it is straightforward since $f_\theta(x_t, t, s) = a(t, s)x_t + b(t, s)F_\theta(x_t, t, s)$ is a neural solution function to the velocity ODE, we only need to sample $x_1 \sim \mathcal{N}(0, I)$ and apply this operator with $t = 1$ and $s = 0$ to obtain the clean data in a single step. Furthermore, by selecting different t and s, our model can also achieve multi-step sampling.

5 EXPERIMENTS

5.1 SETTINGS

We conduct our major experiments on the ImageNet 256×256 dataset (Deng et al., 2009). SoFlow models operate within the latent space of a pre-trained VAE [1](Rombach et al., 2022), which is a common practice in recent works (Peebles & Xie, 2023b; Geng et al., 2025). The tokenizer converts 256×256 images into a 32×32×4 latent representation. We assess generation quality using the Fréchet Inception Distance (FID) (Heusel et al., 2017), computed over a set of 50,000 generated samples. To evaluate computational efficiency, we report the number of function evaluations (NFE), with a particular focus on the single-step (1-NFE) scenario. All models presented were trained from scratch. In addition to the standard time variable t, our model incorporates an additional time variable s. We provide this to the networks by feeding the positional embeddings (Vaswani et al., 2017) of their difference, $s - t$. For more implementation details, please refer to Appendix A.

5.2 ABLATION STUDY

Table 1: **Ablation studies on ImageNet 256×256 class-conditional generation.** All models have 131M parameters and are trained with a batch size of 256 for 400K iterations from scratch.

$l \to t$ schedule	FID-50K (1-NFE)
Exponential	**59.97**
Cosine	60.10
Linear	60.64
Constant	60.13

(a)

flow matching data ratio	FID-50K (1-NFE)
0%	66.55
25%	61.53
50%	60.24
75%	**59.97**

(b)

loss weighting coefficient p	FID-50K (1-NFE)
0.0	69.11
0.5	60.59
1.0	**59.97**
1.5	64.25

(c)

noising schedule & parameterization	FID-50K (1-NFE)
Linear, Euler	**12.89**
Linear, Triangular	23.57
Triangular, Euler	14.26
Triangular, Triangular	19.87

(d)

CFG guidance strength w	FID-50K (1-NFE)
1.5	35.70
2.0	22.37
2.5	15.68
3.0	**12.89**

(e)

velocity mix ratio m	FID-50K (1-NFE)
0.25	**12.89**
0.5	15.67
0.75	17.18
1.0	19.18

(f)

We conduct various ablation studies to determine the optimal hyperparameters for SoFlow models. Following the methodology of Geng et al. (2025), we utilize the DiT-B/4 architecture in our experiments, which features a "base" sized diffusion transformer with a patch size of 4. We train our models for 400K iterations. For performance reference, the original DiT-B/4 (Peebles & Xie, 2023b) achieved an FID-50K of 68.4 with 250-NFE sampling. Meanflow (Geng et al., 2025) reports an FID-50K of 61.06 with 1-NFE sampling, and they claim to have reproduced SiT-B/4 (Ma et al., 2024) with an FID-50K of 58.9 using 250-NFE sampling. The FID-50K scores mentioned here are without classifier-free guidance (CFG). Our ablation studies are conducted in two groups: a first group of three experiments without classifier-free guidance (CFG) and a second group of three with CFG. Table 1 presents our results, which are analyzed as follows:

$l \to t$ **Schedule** We compare exponential, cosine, linear, and constant schedules in our experiments (Table 1a). The similar FID-50K results show that the speed of the $l \to t$ transition during the training process has a relatively small influence on performance. We choose the exponential schedule following previous works (Song & Dhariwal, 2023; Geng et al., 2024).

[1]SD-VAE: `https://huggingface.co/stabilityai/sd-vae-ft-mse`

Flow Matching Data Ratio Although the solution consistency loss alone is sufficient to enable one-step generation (corresponding to a 0% ratio in Table 1b), experimental results show that incorporating a large ratio of flow matching loss is effective for improving performance.

Loss Weighting Coefficient p We observe that the choice of loss metric has a significant influence on the performance of one-step generation, which aligns with previous works (Song & Dhariwal, 2023; Geng et al., 2024; 2025). As shown in Table 1c, the original mean square loss with $p = 0$ performs much worse than $p = 0.5$ or $p = 1$ situations.

Noising Schedule and Parameterization Our method is compatible with different flow matching noising schedules and solution function parameterizations. As presented in Table 1d, experiments show that the linear noising schedule $x_t = (1-t)x_0 + tx_1$ along with the Euler parameterization $f_\theta(x_t, t, s) = x_t + (s-t)F_\theta(x_t, t, s)$ performs best, compared to the triangular noising schedule $x_t = \cos(\frac{\pi}{2}t)x_0 + \sin(\frac{\pi}{2}t)x_1$ and its corresponding parameterization $f_\theta(x_t, t, s) = \cos\frac{\pi}{2}(s-t)x_t + \sin\frac{\pi}{2}(s-t)F_\theta(x_t, t, s)$.

CFG Guidance Strength w Unlike diffusion models, our model enables CFG during the training stage rather than the inference stage, which allows for image generation with 1 NFE. Experiments in Table 1e demonstrate that CFG can also greatly improve the generation quality for our model.

Velocity Mix Ratio m Experiments (Table 1f) show that a relatively small m is beneficial for performance. This can be explained by the fact that the CFG guidance strength w amplifies the randomness in the velocity term, and a smaller m can suppress this randomness by partially replacing the velocity term with the model's prediction of the guided velocity field.

5.3 Comparing with Other Works

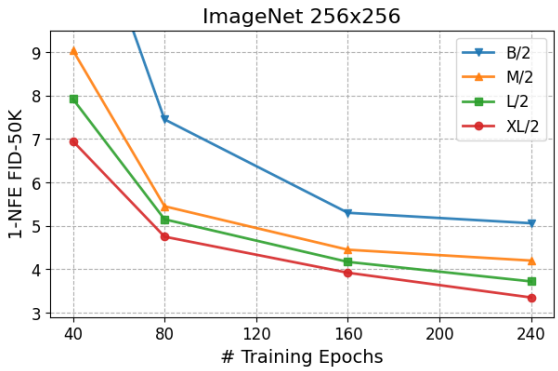

Figure 3: **FID-50K performance of our models.** The plot shows the FID-50K performance of our models with varying numbers of parameters, all trained from scratch on the ImageNet 256×256 dataset. We applied CFG during training and report the FID-50K scores of generated images using a 1-NFE sampling process. The results consistently demonstrate that the performance of our models improves as the model size increases.

ImageNet 256×256 Results We begin by analyzing the 1-NFE FID-50K results of our model across various model sizes and training durations, as shown in Figure 3. Our model's 1-NFE performance gradually improves with an increase in model parameters, which is consistent with previous observations on diffusion transformers (Peebles & Xie, 2023b; Ma et al., 2024).

Next, we compare our model's one-step generation performance against previous models, with the results summarized in Table 2. To make relatively fair comparisons, we train our model with the same batch size and number of iterations as the Meanflow models (Geng et al., 2025). Experimental results show that our model consistently outperforms Meanflow models across all evaluated model sizes when trained from scratch. Specifically, for smaller models with DiT-B/2 and DiT-M/2 architectures, our model demonstrates significant improvements, achieving FID-50K scores of 5.06 and 4.20, respectively. Our model also achieves superior FID-50K values for larger architectures, namely DiT-L/2 and DiT-XL/2, reaching 3.72 and 3.35, respectively. Notably, our models employ Classifier-Free Guidance (CFG) during training, which enables generation with exactly 1-NFE during inference, similar to Meanflow models.

method	epochs	params	NFE	**FID↓**
Generative Adversarial Networks				
BigGAN (Brock et al., 2018)	-	112M	1	6.95
StyleGAN-XL (Sauer et al., 2022)	-	166M	1	**2.30**
GigaGAN (Kang et al., 2023)	-	569M	1×2	3.45
Masked and Autoregressive Models				
Mask-GIT (Chang et al., 2022)	555	227M	8	6.18
MagViT-v2 (Yu et al., 2023)	1080	307M	64	1.78
LlamaGen-XL (Sun et al., 2024)	300	775M	576	2.62
VAR (Tian et al., 2024)	350	2.0B	10	1.80
MAR (Li et al., 2024)	800	943M	64	**1.55**
RandAR-XL (Pang et al., 2025)	300	775M	256	2.22
Multi-step Diffusion Models				
LDM-4-G (Rombach et al., 2022)	170	395M	250×2	3.60
MDTv2 (Gao et al., 2023)	700	676M	250×2	1.63
DiT-XL (Peebles & Xie, 2023a)	1400	675M	250×2	2.27
SiT-XL (Ma et al., 2024)	1400	675M	250×2	2.06
FlowDCN-XL (Wang et al., 2024b)	400	675M	250×2	2.00
SiT-REPA-XL (Yu et al., 2024)	800	675M	250×2	**1.42**
Few-step Consistency Models				
iCT-XL† (Song & Dhariwal, 2023)	-	675M	1×2	20.30
Shortcut-XL (Frans et al., 2024)	250	675M	1×2	10.60
IMM-XL (Zhou et al., 2025)	3840	675M	1×2	7.77
MeanFlow-B (Geng et al., 2025)	240	131M	1	6.17
MeanFlow-M (Geng et al., 2025)	240	308M	1	5.01
MeanFlow-L (Geng et al., 2025)	240	459M	1	3.84
MeanFlow-XL (Geng et al., 2025)	240	675M	1	3.43
SoFlow-B	240	131M	1	5.06
SoFlow-M	240	308M	1	4.20
SoFlow-L	240	459M	1	3.72
SoFlow-XL	240	675M	1	**3.35**

Table 2: **FID-50K results for class-conditional generation on ImageNet-256×256.** ×2 denotes an NFE of 2 per sampling step incurred by CFG. † Results as reported in (Zhou et al., 2025).

CIFAR-10 Results In Table 3, we report unconditional generation results on the CIFAR-10 dataset, where performance is measured by the FID-50K metric with 1-NFE sampling. For our model, we adopt the U-Net architecture (Ronneberger et al., 2015) developed from (Song et al., 2020), aligning with prior works. Our method is applied directly to the pixel space, with a resolution of 32×32. For more implementation details, please refer to Appendix A. Our method achieves competitive performance compared to prior approaches on this dataset.

method	NFE	FID
iCT (Song & Dhariwal, 2023)	1	**2.83**
ECT (Geng et al., 2024)	1	3.60
sCT (Lu & Song, 2024)	1	2.97
IMM (Zhou et al., 2025)	1	3.20
MeanFlow (Geng et al., 2025)	1	2.92
SoFlow	1	2.90

Table 3: **FID-50K results for unconditional generation on CIFAR-10.**

6 Conclusion

We have presented SoFlow, a simple yet effective framework for one-step generative modeling. Our approach directly learns the solution function of the velocity ODE, enabling single-step sampling without iterative solvers. By leveraging a bi-time formulation and a hybrid training objective combining a flow matching and a solution consistency loss, SoFlow naturally support CFG during training and avoid JVP that are not well-optimized on deep learning frameworks like PyTorch. Our method demonstrates competitive performance on class-conditioned Imagenet 256×256 generation task, outperforming Meanflow models when trained from scratch under the same settings.

REFERENCES

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video generation models as world simulators. 2024. *URL https://openai. com/research/video-generation-models-as-world-simulators*, 3:1, 2024.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. 2009.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.

Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Mdtv2: Masked diffusion transformer is a strong image synthesizer. *arXiv preprint arXiv:2303.14389*, 2023.

Zhengyang Geng, Ashwini Pokle, and J Zico Kolter. One-step diffusion distillation via deep equilibrium models. *Advances in Neural Information Processing Systems*, 36:41914–41931, 2023.

Zhengyang Geng, Ashwini Pokle, William Luo, Justin Lin, and J Zico Kolter. Consistency models made easy. *arXiv preprint arXiv:2406.14548*, 2024.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Jonathan Heek, Emiel Hoogeboom, and Tim Salimans. Multistep consistency models. *arXiv preprint arXiv:2403.06807*, 2024.

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. 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10124–10134, 2023.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. *arXiv preprint arXiv:2310.02279*, 2023.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37: 56424–56445, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. *arXiv preprint arXiv:2410.11081*, 2024.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023a.

Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. *Advances in Neural Information Processing Systems*, 36:76525–76546, 2023b.

Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pp. 23–40. Springer, 2024.

Ziqi Pang, Tianyuan Zhang, Fujun Luan, Yunze Man, Hao Tan, Kai Zhang, William T Freeman, and Yu-Xiong Wang. Randar: Decoder-only autoregressive visual generation in random orders. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 45–55, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. 2023a.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023b.

Yansong Peng, Kai Zhu, Yu Liu, Pingyu Wu, Hebei Li, Xiaoyan Sun, and Feng Wu. Flow-anchored consistency models. *arXiv preprint arXiv:2507.03738*, 2025.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

A Polyak, A Zohar, A Brown, A Tjandra, A Sinha, A Lee, A Vyas, B Shi, CY Ma, CY Chuang, et al. Movie gen: A cast of media foundation models, 2025. *URL https://arxiv. org/abs/2410.13720*, pp. 51, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation. *arXiv preprint arXiv:2506.14603*, 2025.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 conference proceedings*, pp. 1–10, 2022.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pp. 87–103. Springer, 2024.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*, 2023.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Fu-Yun Wang, Zhengyang Geng, and Hongsheng Li. Stable consistency tuning: Understanding and improving consistency models. *arXiv preprint arXiv:2410.18958*, 2024a.

Shuai Wang, Zexian Li, Tianhui Song, Xubin Li, Tiezheng Ge, Bo Zheng, and Limin Wang. Exploring dcn-like architecture for fast image generation with arbitrary resolution. *Advances in Neural Information Processing Systems*, 37:87959–87977, 2024b.

Zidong Wang, Yiyuan Zhang, Xiaoyu Yue, Xiangyu Yue, Yangguang Li, Wanli Ouyang, and Lei Bai. Transition models: Rethinking the generative learning objective. *arXiv preprint arXiv:2509.04394*, 2025.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6613–6623, 2024.

Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, et al. Language model beats diffusion–tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

Linqi Zhou, Stefano Ermon, and Jiaming Song. Inductive moment matching. *arXiv preprint arXiv:2503.07565*, 2025.

Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In *Forty-first International Conference on Machine Learning*, 2024.

A Implementation Details

We first provide a detailed description of the different scheduling strategies used to determine the intermediate time variable l during training. Recall that l is computed from t and s as follow:

$$l = t + (s - t) \times r(k, K),$$

where k, K represent the current and total training steps, and the function $r(k, K)$ is a monotonically decreasing function that controls the progression of l from an initial value near t towards t itself as training advances. Below we specify the four schedules compared in our ablation studies:

Exponential schedule:

$$r(k, K) = r_{\text{init}} \times \left(\frac{r_{\text{end}}}{r_{\text{init}}}\right)^{\frac{k}{K}},$$

Cosine schedule:

$$r(k, K) = r_{\text{end}} + (r_{\text{init}} - r_{\text{end}}) \times \frac{1}{2}\left(1 + \cos\left(\pi \cdot \frac{k}{K}\right)\right),$$

Linear schedule:

$$r(k, K) = r_{\text{init}} + (r_{\text{end}} - r_{\text{init}}) \times \frac{k}{K},$$

Constant schedule:

$$r(k, K) = r_{\text{end}}.$$

In all cases, l is clamped to satisfy $l < t - 10^{-4}$ to ensure numerical stability, s is also clamped to satisfy $s < t - 10^{-4}$ at the same time. The initial value r_{init} and end value r_{end} are hyperparameters controlling the starting and final relative position between l and t.

We now detail the training configurations for our model on two benchmark datasets. All experiments are run on NVIDIA H100 GPUs.

ImageNet 256×256 We employ the Adam optimizer (Kingma & Ba, 2014) with a constant learning rate of 1×10^{-4} and betas set to $(0.9, 0.99)$, without learning rate decay or weight decay. Following standard practice, we evaluate model performance using Exponential Moving Average (EMA) with a decay rate of 0.9999. For time sampling, the logit-normal distribution parameters are set as: $\mu_{\text{FM}} = -0.2, \sigma_{\text{FM}} = 1.0$; $\mu_t = 0.2, \sigma_t = 0.8$; $\mu_s = -0.2, \sigma_s = 0.8$. The flow matching data ratio is 75% and the adaptive loss weight coefficient $p = 1.0$. The schedule parameters are $r_{\text{init}} = \frac{1}{10}$ and $r_{\text{end}} = \frac{1}{320}$. We use a linear noising schedule with Euler parameterization. The CFG strength w and velocity mix ratio m are set to 3.0 and 0.25, respectively. Following common practice, we linearly decay the CFG strength to 0 gradually for high-noise areas, where $t > 0.8$. Finally, architectural details are provided in Table 4.

Table 4: **Architectural configurations on ImageNet 256×256.**

architectures	B/4	B/2	M/2	L/2	XL/2
parameters (M)	131	131	308	459	676
FLOPs (G)	5.6	23.1	54.0	119.0	119.0
depth	12	12	16	24	28
hidden dimension	768	768	1024	1024	1152
attention heads	12	12	16	16	16
patch size	4×4	2×2	2×2	2×2	2 × 2
training epochs	80	240	240	240	240

CIFAR-10 Training uses a batch size of 1024 for 800K iterations, consistent with MeanFlow. We adopt RAdam (Liu et al., 2019) with a learning rate of 1×10^{-4}, following (Song & Dhariwal, 2023; Geng et al., 2024). Time sampling parameters are: $\mu_{\text{FM}} = -0.4, \sigma_{\text{FM}} = 1.7$; $\mu_t = -0.6, \sigma_t = 1.5$; $\mu_s = -4.0, \sigma_s = 1.5$. The flow matching data ratio is 25% with $p = 0.5$. Schedule parameters are set to smaller values than ImageNet: $r_{\text{init}} = \frac{1}{50}$ and $r_{\text{end}} = \frac{1}{2500}$. Linear noising schedule and Euler parameterization are used.

B MORE VISUAL SAMPLES

Figure 4: We show curated samples from class-conditioned generation by the DiT-XL/2 model with 1-NFE on the ImageNet 256×256 dataset.

Figure 5: We show uncurated samples from unconditional generation by the UNet model with 1-NFE on the CIFAR-10 dataset.

C LARGE LANGUAGE MODEL USAGE

We only adopt large language models to polish the writing.