ENHANCING SHORTCUT MODELS WITH CUMULATIVE SELF-CONSISTENCY LOSS FOR ONE-STEP DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Although iterative denoising (i.e., diffusion/flow) methods offer strong generative performance, they suffer from low generation efficiency, requiring hundreds of steps of network forward passes to simulate a single sample. Mitigating this requires taking larger step-sizes during simulation, thereby allowing one- or few-step generation. Recently proposed shortcut model learns larger step-sizes by enforcing alignment between its direction and the path defined by a base many-step flow-matching model through a self-consistency loss. However, its generation quality is significantly lower than the base model. In this paper, we interpret the self-consistency loss through the lens of optimal control by formulating the few-step generation as a controlled base generative process. This perspective enables us to develop a general cumulative self-consistency loss that penalizes the misalignment at both the current step and future steps along the trajectory. This encourages the model to take larger step-sizes that not only align with the base model at the current time step but also guide subsequent steps towards high-quality generation. Furthermore, we draw a connection between our approach and reinforcement learning, potentially opening the door to a new set of approaches for few-step generation. Extensive experiments show that we significantly improve one- and few-step generation quality under the same training budget.

1 INTRODUCTION

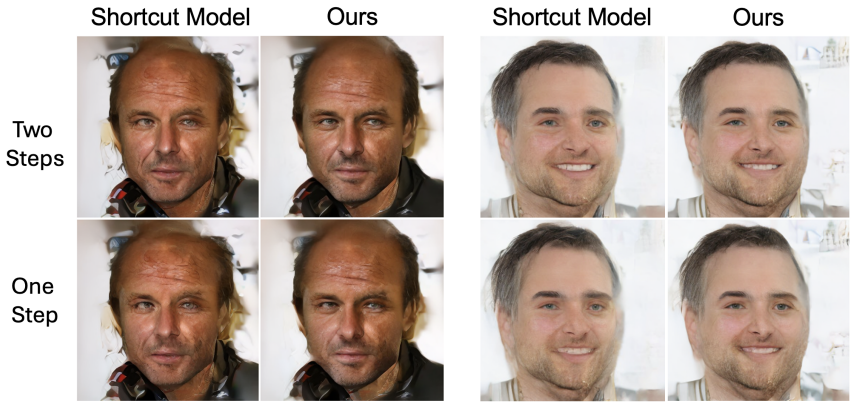

Figure 1: Two-step and one-step image generation using shortcut models (Frans et al., 2025) and our proposed method on CelebA-256 dataset. Our method generates images with less artifacts and higher sharpness than the shortcut model.

Diffusion (Song & Ermon, 2019; Ho et al., 2020; Kingma et al., 2021; Song et al., 2021) and flow-matching (Lipman et al., 2023; Albergo et al., 2023) models have demonstrated remarkable capabilities in generating high-quality images (Rombach et al., 2022; Luo et al., 2023a; Esser et al., 2024), video (Esser et al., 2023; Gupta et al., 2024; Luo et al., 2023b), audio (Huang et al., 2023; Liu et al., 2023a; Kong et al., 2021), and molecular graphs (Vignac et al., 2023; Jo et al., 2022; Eijkelboom et al., 2024). The generation involves iteratively transforming random noise into structured data,

typically requiring hundreds of steps of forward passes through a neural network. This renders the process inefficient and computationally expensive. It is a key limitation in comparison to single-step generative models, such as VAE (Kingma & Welling, 2014; Vahdat & Kautz, 2020), GAN (Goodfellow et al., 2014), and normalizing flows (Dinh et al., 2017; Kingma & Dhariwal, 2018).

To achieve one- or few-step generation, previous efforts distill knowledge from a pretrained diffusion model into an efficient student model. A way for one-step generation typically involves training a student model to learn a direct mapping from noise-image pairs generated by the pre-trained model (Luhman & Luhman, 2021; Zhao et al., 2023; Zheng et al., 2023; Yin et al., 2024). A set of approaches avoid the cost of generating such pairs, and support flexible generation budgets by allowing few-step generation (Berthelot et al., 2023; Ghimire et al., 2023; Gu et al., 2023; Liu et al., 2023b; 2024; Salimans & Ho, 2022; Song et al., 2023). These methods either progressively distill knowledge by halving the number of generation steps at each stage (Salimans & Ho, 2022), or enforce straighter flow paths during distillation (Liu et al., 2023b; Lee et al., 2024). These distillation-based approaches require two training phases: training a base diffusion model and distilling its knowledge into a student model. Consistency models propose a single-phase training for few-step generation by learning the transformation of each noisy sample in the generation trajectory to the same final output (Song et al., 2023; Song & Dhariwal, 2024). However, they suffer from poor training stability and bias introduced by discretization (Frans et al., 2025). Recently, shortcut models, a family of models with step-sizes larger than that of the base models, show promising performance by conditioning the model's output not only on time-steps and noisy inputs but also on the simulation step-sizes (Frans et al., 2025). This simple but elegant design allows for generation under a specified budget simply by conditioning the model on its corresponding step size. They jointly train base model and the shortcut models with a standard flow-matching loss and a self-consistency loss (SL), which aligns the shortcut model's outputs with the base model at each time step. Although the experiments show improved few-step generation performance, it still lags significantly behind the base flow-matching models, emphasizing the need for further improvement. Moreover, the introduction of the SL loss in the paper lacks a theoretical foundation.

In this paper, we focus on improving shortcut models by identifying limitation in the SL loss, and introducing an improved version. Specifically, by formulating the shortcut models as a controlled version of the base generative models, we propose to interpret the SL loss through the lens of an optimal control framework (Bryson & Ho, 1975). Through this perspective, we show that the SL loss corresponds to a special case of the optimal control objective, since it penalizes misalignment between the shortcut and the base model only at the current time step, overlooking the future ramifications of this immediate deviation. By drawing upon the concept of expected future cost in a controlled process, we thus propose a general cumulative self-consistency loss (CSL), which penalizes both immediate and future misalignments cumulatively over time. Moreover, we establish a connection between our algorithm and on-policy reinforcement learning methods (Mnih et al., 2013; Lillicrap et al., 2015) by interpreting the expected future cost as a value function. Similar to on-policy algorithms aiming to maximize an agent's cumulative reward rather than optimizing for immediate gains, our approach optimizes the cumulative cost along the generation trajectory. Evaluating the generative performance on CelebA256 and CIFAR10 datasets shows that our method robustly outperforms the state-of-the-art single-phase training approaches with the same training budget. Our contributions can be summarized as follows:

- We propose a novel perspective that frames few-step generation as a controlled process, and interpret the SL loss as a special case of an optimal control objective.
- We propose a novel CSL loss that penalizes both immediate and future misalignments between the few-step mode and the base generative model cumulatively over time.
- Training with the proposed CSL loss significantly outperforms shortcut models and other few-step generation baselines on benchmark datasets.

2 Related Works

2.1 Few-Step Generation

Distillation-based approaches aim to transfer knowledge from a pretrained model to a student model (Luhman & Luhman, 2021; Zhao et al., 2023; Zheng et al., 2023; Yin et al., 2024; Liu et al., 2023b;

Salimans & Ho, 2022). Some methods allow only fixed-step generation or require generating large volumes of noise-image pairs (Luhman & Luhman, 2021; Zhao et al., 2023; Zheng et al., 2023; Yin et al., 2024; Liu et al., 2023b), while others reduce steps via bootstrapping but demand retraining a new model for each step reduction (Salimans & Ho, 2022). Overall, these approaches are either computationally expensive or inefficient when scaling to different step counts. Moreover, all these approaches follow a two-phase pipeline: pretraining a base model followed by distilling knowledge into few-step models. In contrast, consistency models (Song et al., 2023; Song & Dhariwal, 2024; Lu & Song, 2025) introduce a single-phase training strategy that trains a model to map noisy inputs from any time step directly to the final denoised version, essentially enforcing transformation consistency along the denoising trajectory. Remarkably, this class of models achieve one- or few-step performance close to the base diffusion or flow-matching model. However, they are complicated by design and naturally unstable while training, requiring substantial engineering efforts like adaptive weighing for training stability Lu & Song (2025). Recently, Frans et al. (2025) proposed shortcut models for few-step generation, featuring a simple, intuitive design and a straightforward single-phase training recipe, making them accessible for real-world deployment. Shortcut models condition not only on the noisy input and time step but also on the step size. By learning a family of denoising functions conditioned on the step size, they provide a flexible and efficient framework for few-step generation using a single model across different computation budgets. The SL loss is employed to align the generation trajectory of the model at larger step sizes with that of the base model, yielding strong performance. Nevertheless, the performance of few-step models still lags considerably behind that of the base model.

In this work, we focus on improving shortcut models. Since the original formulation of the SL loss lacks a theoretical foundation, we establish one, which allows us to identify its limitations and propose an improved variant. In addition, our theoretical framework uncovers a previously unidentified connection between few-step generation and reinforcement learning.

2.2 Leveraging Control Theory to Guide Diffusion-Based Generation

A few works have utilized control-theoretic principles to steer or fine-tune base diffusion and flow-matching models (Wang et al., 2025; Sprague et al., 2024; Domingo-Enrich et al., 2025). OC-Flow (Wang et al., 2025) leverages an optimal control framework to guide pre-trained flow-matching models for controlled generation. Domingo-Enrich et al. (Domingo-Enrich et al., 2025) introduce a fine-tuning algorithm that interprets reward-based optimization through the lens of stochastic optimal control.

3 Background

3.1 Optimal Control

Let X_t^u denote the state of a controlled dynamical system at time t defined by:

$$dX_t^u = \big(b(X_t^u, t) + \sigma(t)\, u_\theta(X_t^u, t)\big)\, dt, \quad X_0^u \sim p_0, \tag{1}$$

where $b(X_t^u, t)$ is the base drift, $u_\theta(X_t^u, t)$ is the drift, called the control vector field, $\sigma(t)$ is a control coefficient, and p_0 is the initial distribution. These jointly define the evolution of the controlled process X_t^u. Optimal control problem aims to minimize the following objective:

$$\min_{u_\theta \in \mathcal{U}} \int_0^1 \Big(f\big(u_\theta(X_t^u, t), t\big) + g(X_t^u, t)\Big)\, dt + h(X_1^u), \tag{2}$$

where $f(u_\theta(X_t^u, t), t)$ penalizes the magnitude of the control, $g(X_t^u, t)$ is the intermediate cost, and $h(X_1^u)$ is the terminal cost. We also define the cost functional $J(u; x, t')$, representing the expected future cost starting from state x at time t':

$$J(u_\theta; x_{t'}, t') = \int_{t'}^1 \Big(f\big(u_\theta(X_t^u, t), t\big) + g(X_t^u, t)\Big)\, dt + h(X_1^u), \quad x_{t'} = X_{t'}^u \tag{3}$$

A result from optimal control, which we will later use in our analysis, is the expression for the derivative of $J(u_\theta; x_{t'}, t')$ with respect to the control parameters θ is:

$$\frac{dJ(u_\theta; x_{t'}, t')}{d\theta} = \int_{t'}^1 \frac{\partial}{\partial \theta} f\big(u_\theta(X_t^u, t), t\big)\, dt + \int_{t'}^1 \frac{\partial u_\theta(X_t^u, t)}{\partial \theta} \sigma(t) \nabla_{x_t} J(u_\theta; x_t, t) dt \tag{4}$$

3.2 FLOW MATCHING

Flow-matching models (Lipman et al., 2023; Albergo et al., 2023) formulate data generation as simulating an ordinary differential equation (ODE), iteratively converting random noise into realistic data samples. Given a model with parameter θ that outputs velocity field v_θ, the denoising process is represented as:

$$dx_t = v_\theta(x_t, t)\, dt\ , \quad t \in [0, 1],\ x_0 \sim \mathcal{N}(0, I)$$

The goal is to train the model such that the distribution of samples at $t = 1$ matches the data distribution. We consider the optimal transport formulation of flow-matching (Lipman et al., 2023) in which the model is trained to optimize the following objective:

$$\mathcal{L}_{FM}(\theta) = \mathbb{E}_{x_0,x_1}\left[\|v_\theta(x_t, t) - v_t\|^2\right]; \quad x_t = (1-t)x_0 + tx_1, \quad v_t = x_1 - x_0 \tag{5}$$

Here, x_1 is a sample from the data distribution, $x_1 \sim D$, and v_t represents the direction from noise to the data sample. When both x_0 and x_1 are known, v_t is deterministic. However, multiple (x_0, x_1) pairs may correspond to the same x_t, making v_t inherently stochastic. When trained with the objective in Equation 5, flow-matching models learn the expected velocity conditioned on x_t, denoted as $\mathbb{E}[v_t|x_t]$.

3.3 SHORTCUT MODELS

The trajectory defined by the flow-matching ODE is typically curved, necessitating small step sizes for accurate simulation, resulting in inefficient generation. Using larger step sizes to enable faster, few-step generation leads to deviations from the true trajectory and significantly degrades sample quality, often causing failure when generating with fewer than four steps. To overcome this limitation, Frans et al. (Frans et al., 2025) propose shortcut models, which learn to take larger steps while accounting for the future curvature of the trajectory, thereby avoiding deviation from the original path. This approach enables few-step generation with improved sample quality. Notably, both the flow-matching and shortcut models are parameterized by the same backbone network, with different variants conditioned on the step size. A step taken by the shortcut model is given by:

$$x_{t+d} = x_t + s(x_t, t, d)\ .d \tag{6}$$

where d is the step size, and $s(x_t, t, d)$ denotes the normalized direction from x_t to the point x_{t+d} on the trajectory. Shortcut models of different step sizes d can be trained on the same backbone network. The flow-matching model is the one conditioned with $d = 0$ i.e., $s(x_t, t, 0)$. Shortcut models are trained to enforce consistency across step sizes: taking a single step of size $2d$ should be equivalent to taking two sequential steps of size d.

$$s(x_t, t, 2d) = s_{\text{target}}, \quad s_{\text{target}} = \frac{s(x_t, t, d)}{2} + \frac{s(x'_{t+d}, t+d, d)}{2},$$
$$x'_{t+d} = x_t + s_\theta(x_t, t, d)\ .d$$

The target for the shortcut model $s(x_t, t, 2d)$ is thus generated by bootstrapping. The joint training objective of the flow-matching and shortcut models is:

$$\mathcal{L}_{\text{shortcut}}(\theta) = \mathbb{E}_{x_0,x_1,t,d}\Big[\underbrace{\|s_\theta(x_t, t, 0) - (x_1 - x_0)\|^2}_{\text{Flow-Matching}} + \underbrace{\|s_\theta(x_t, t, 2d) - s_{\text{target}}\|^2}_{\text{Self-Consistency}}\Big] \tag{7}$$

4 CONTROL THEORETIC FORMULATION OF FEW-STEP MODELS

Both the forward and backward processes in flow-matching and diffusion models are governed by the Fokker–Planck equation, which describes the evolution of a probability density over time (Risken, 1996). From this perspective, one- or few-step generation can be viewed as training a neural network to approximate the solution of an underlying ODE. Consequently, when training a shortcut model to approximate this solution, the resulting integration error is cumulative in nature. To formalize this intuition, we model the output of the shortcut model itself as the solution to an ODE, defined as follows:

$$dX_t^u = [b(X_t^u, t) + u_\theta(X_t^u, t)]dt \tag{8}$$

where, X_t^u denotes the output of a shortcut model trying to match the output from a flow-matching base process $dX_t = b(X_t, t)\ dt$. This formulation implies that the shortcut model's output incurs an error u_θ at each point in time t. Consequently, the error at time t reflects the cumulative effect of all errors at earlier time points $\tau < t$. Another advantage of this formulation is that in Equation (8) provides a new perspective of viewing the few-step model as a controlled process. Unlike the standard optimal control formulation, where a control vector field u_θ is introduced to steer the base process toward optimizing a predefined objective (e.g. Equation 3), here u_θ is an unknown error that is implicitly introduced along the generation trajectory of a shortcut model. Our goal is to minimize its magnitude to ensure that the few-step models closely align with the base model. Accordingly, we define an objective by setting $g = h = 0$ in Equation 3.

$$J(u_\theta; x_{t'}, t') = \int_{t'}^{1} f\big(u_\theta(X_t^u, t), t\big)\ dt \tag{9}$$

The future cost, J, thus becomes the primary quantity of interest. It is important to note that if $u_\theta = 0$ for all t and X, the controlled process matches exactly with the base process, and the few-step method reduces to the base flow-matching model. Next, we show how Equation 9 generalizes the SL loss.

4.1 Self-Consistency Loss as an Optimal Control Objective

Lemma 1 *Let $f\big(u_\theta(X_t^u, t), t\big) = \|u_\theta(X_t^u, t)\|^2 \boldsymbol{\delta}(t' - t)$, where $\boldsymbol{\delta}(t)$ is the dirac delta function. The objective in Equation 9 becomes:*

$$J_{SL}(u_\theta; x_{t'}, t') = \int_{t'}^{1} \|u_\theta(X_t^u, t)\|^2 \boldsymbol{\delta}(t' - t)dt = \|u_\theta(x_{t'}, t')\|^2\ , \quad x_{t'} = X_{t'}^u \tag{10}$$

From Equation 10, if we consider d-step model as shortcut model $\frac{dX_t^u}{dt}$ and $2d$-step model as base drift $b(X_t^u, t)$, then for the discrete case, u_θ is the difference between the average of the two steps taken by the base model and a single step taken by the shortcut model: $u_\theta(x_t, t) = s_\theta(x_t, t, 2d) - s_{\text{target}}$. The optimal control objective equals to the self-consistency loss term:

$$J_{SL}(u_\theta; x_t, t) = \|s_\theta(x_t, t, 2d) - s_{\text{target}}\|^2$$

5 Cumulative Self-Consistency Loss (CSL)

In Lemma 1, the value of $f\big(u_\theta(X_t^u, t)\big)$ is non-zero only at time t', due to the presence of the Dirac delta function. As a result, the objective J_{SL} accounts solely for the error u_θ at the current step, ignoring all future errors. To incorporate the errors along the entire trajectory, we relax the delta function constraint and propose the following two loss functions:

Lemma 2 *Let, $f\big(u_\theta(X_t^u, t), t\big) = \|u_\theta(X_t^u, t')\|^2\ , \forall t > t'$, the objective in Equation 9 becomes:*

$$J_{USL}(u_\theta; x_{t'}, t') = \int_{t'}^{1} \|u_\theta(X_t^u, t)\|^2 dt = (1 - t')\ \|u_\theta(x_{t'}, t')\|^2\ , \quad x_{t'} = X_{t'}^u \tag{11}$$

J_{USL} consider errors over the entire time $t' < t < 1$, where USL stands for Uniform Self-Consistency Loss. However, it makes a naive assumption that the error u_θ is uniform over the entire trajectory. We further relax the uniformity assumption and propose a CSL objective.

Lemma 3 *Let $f\big(u_\theta(X_t^u, t), t\big) = \|u_\theta(X_t^u, t)\|^2\ , \forall t > t'$, the objective in Equation 9 becomes:*

$$J_{CSL}(u_\theta; x_{t'}, t') = \int_{t'}^{1} \|u_\theta(X_t^u, t)\|^2 dt\ , \quad x_{t'} = X_{t'}^u \tag{12}$$

5.1 Analysis and Comparison with Self-Consistency Loss

Using 4, the gradient of J_{CSL} with respect to the parameters θ is given as (assuming $\sigma(t) = 1$):

$$\frac{dJ_{CSL}(u_\theta; x_{t'}, t')}{d\theta} = \int_{t'}^{1} \Big(2u_\theta + \nabla_{x_t} J_{CSL}(u_\theta; x_t, t)\Big) \frac{\partial u_\theta}{\partial \theta} dt$$

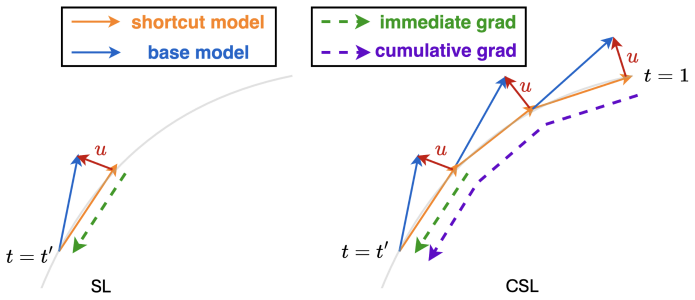

Figure 2: Illustration of the SL (left) and CSL (right) losses. The solid blue, orange, and red arrows denote the direction of the base model, the shortcut model, and the error u_θ, respectively. The error is the difference between the direction taken by the base model and the shortcut model. For SL, we calculate the error only at the current time t', while for CSL, we calculate and aggregate the error at every point from $t = t'$ to the final time $t = 1$. CSL allows the model to learn from cumulative gradient in addition to the immediate gradient as denoted by dashed purple and green arrows respectively.

Analyzing the integrand,

$$\left(2u_\theta + \nabla_{x_t} J_{CSL}(u_\theta; x_t, t)\right) \frac{\partial u_\theta}{\partial \theta} = \Bigg(\underbrace{2u_\theta}_{\text{immediate grad}} + \underbrace{\nabla_{x_t} \int_t^1 \|u_\theta(X_t^u, t)\|^2 dt}_{\text{cumulative grad}} \Bigg) \frac{\partial u_\theta}{\partial \theta} \tag{13}$$

In Equation 13, the gradient of J_{CSL} comprises two components: an immediate gradient term corresponding to the error at the current step, and a cumulative gradient term corresponding to the cumulative error along the trajectory. In contrast, the gradient of the J_{SL} includes only the immediate gradient. Note that the cumulative gradient is a gradient of the future errors with respect to the state x_t. Optimizing J_{CSL} therefore encourages the shortcut model not only to align with the base model at the current step, but also to guide the trajectory towards a state x_t that facilitates high-quality generation by supporting the alignment in the subsequent steps as well.

5.2 CSL Estimation and Training

Since we are working with discrete time steps, the loss J_{CSL} in Equation 12 can be expressed as a summation of the norm of u_θ at every discrete step along the trajectory:

$$J_{CSL}(u_\theta; x_{nd}, nd) = \sum_{k=n}^{R'} \|u_\theta(x_{dk}, dk, d)\|^2, \quad R' = \frac{1}{d}, \quad n \in \{1, 2, \ldots, R' - 1\} \tag{14}$$

Let $R = R' - n + 1$ denote the number of terms in the summation. Replacing $R' = \frac{1}{d}$ with $R' = n$ (i.e. $R = 1$) results in an SL objective. An illustration of the SL and CSL losses is presented in Figure 2. Estimating summation in Equation 14 requires simulation, which can be computationally demanding. However, we can approximate the summation by including only a few terms. Specifically, we find that using only two terms, i.e., $R = 2$, significantly improves few-step generation performance with negligible computational overhead. The effectiveness of the two-term estimation stems from the fact that, for few-step models with two or four steps, even a two-step simulation covers a substantial portion of the trajectory, covering the full trajectory for a 2-step model and 50% of the trajectory for a 4-step shortcut model. The algorithm for our method is provided in the Appendix.

5.3 Connection to Reinforcement Learning

In our setting, the reinforcement learning analogy can be drawn by viewing the agent as attempting to transform noise into a data sample, where the states correspond to intermediate noisy samples along the generation path, the action is the direction of the next step, and the reward is defined as the negative magnitude of u_θ. Under this view, the objective J_{CSL} in Equation 12 plays a role similar

to a value function: it quantifies the cumulative cost of following a policy along the trajectory. Moreover, since few-step generation involves a small, fixed number of steps, we do not require a separate network to approximate J_{CSL}; instead, it can be estimated directly by rolling out the actions. A more detailed discussion on this connection is provided in the Appendix F.

Table 1: Comparison of FID-50K ↓ scores of the baselines and our method on CelebA-256 and CIFAR10. (* indicates that results are taken from Frans et al. (2025), † indicates that separate models were trained for different generation step settings.)

	CelebA-256				**CIFAR10**			
	128-Step	Four-Step	Two-Step	One-Step	128-Step	Four-Step	Two-Step	One-Step
Two Phase Training								
Reflow	12.80±0.03	13.77±0.05	14.48±0.05	16.07±0.02	13.93±0.05	14.92±0.05	15.59±0.14	16.98±0.06
PD†	7.96±0.07	14.49±0.07	16.73±0.02	20.40±0.15	7.89±0.07	10.75±0.09	11.80±0.08	**13.26±0.09**
Single Phase Training								
FM	7.92±0.04	62.8±0.04	112.1±0.07	321.2±0.04	7.95±0.09	65.03±0.03	177.9±0.12	385.1±0.11
CT*	53.7	19.0	-	33.2	-	-	-	-
ST	**7.83±0.04**	9.36±0.05	12.56±0.02	20.46±0.02	7.37±0.03	9.15±0.13	11.79±0.07	19.80±0.03
ST-USL	7.95±0.02	9.18±0.08	12.00±0.05	19.41±0.03	7.37±0.08	9.35±0.07	11.65±0.08	19.57±0.05
ST-CSL	7.88±0.04	**8.98±0.02**	**10.96±0.02**	**18.37±0.02**	**7.13±0.03**	**8.10±0.09**	**9.24±0.09**	**17.76±0.02**

6 EXPERIMENTS

In this section, we empirically evaluate the proposed CSL. First, to evaluate the generative performance, in section 6.1 we make a comparison of the model's performance with the baselines. In sections 6.2 and 6.3, we assess if the improvement brought by CSL is consistent across varying backbone network size and varying ratio of flow-matching to bootstrap targets ($B : K$) (see algorithm E) along with the training time comparison of the methods. Lastly, in section 6.4 we assess the effect of increasing the number of terms R on the performance. For all evaluations, we use FID-50K score as the comparison metric. We train the models on the CelebA256 Liu et al. (2015) and CIFAR10 Krizhevsky (2009) datasets using NVIDIA A100 and RTX A5000 GPUs.

6.1 PERFORMANCE COMPARISON

We consider two categories of baselines for performance comparison. **1. Two-Phase distillation approaches:** Progressive Distillation (PD) Salimans & Ho (2022), Reflow Liu et al. (2023b). **2. Single-Phase training approaches:** Flow-Matching (FM) Lipman et al. (2023), Consistency Training (CT) Song et al. (2023), Shortcut Models (ST)Frans et al. (2025)

For a fair comparison with the shortcut model, we make sure the training budget is the same for both methods. We use the same number of flow-matching and bootstrap targets per batch. To achieve this, since we create two bootstrap targets per datapoint for $R = 2$, we only make use of $K/2$ datapoints for bootstrap targets in our case, while for the shortcut model we use K datapoints. We require additional compute for the additional network forward pass step during bootstrap targets generation (step 18 in Algorithm E), which adds only 7% of extra computation and training time in practice. Similar to Frans et al. (2025), we use the medium-scale diffusion transformer model DiT-B-2 (Peebles & Xie, 2023) as a backbone network. All the other hyperparameters are the same, and the details are provided in the Appendix. The FID-50K scores for the baselines and our method are reported in Table 1, the base flow-matching model is the one with 128-steps. Our method, trained with J_{USL} and J_{CSL}, are referred to as shortcut-USL (ST-USL) and shortcut-CSL (ST-CSL), respectively. The results show that ST-USL consistently outperforms the baseline ST in one-, two-, and four-step generation, except for the four-step case on CIFAR10. Notably, the performance of ST-CSL surpasses both ST and ST-USL on both datasets across all the steps, with

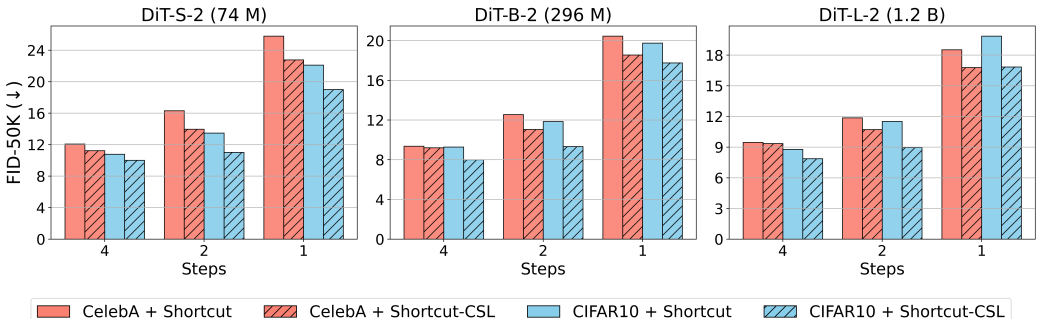

Figure 3: FID score comparison between shortcut and shortcut-CSL (ours) across different backbone network sizes (parameter counts). The networks are diffusion transformers (DiT) of varying sizes. Our model consistently outperforms the shortcut model across all network sizes and generation steps on both CelebA-256 and CIFAR10 datasets.

the exception of the base 128-step model on CelebA, where it achieves performance comparable to ST. This shows the effectiveness of using CSL in improving the generation quality of few-step models. Additionally, while PD achieves superior one-step generation performance on CIFAR-10, and Reflow does so on CelebA, both methods require a two-phase training procedure. Furthermore, they offer a less favorable trade-off between computational cost and performance, underperforming on 2- and 4-step generation tasks compared to our method. PD also requires training and deploying separate models for each generation budget, which is time-inefficient.

Next, we empirically asses the analysis in section 5.1 that our method trains the model to not only minimize current but also future misalignments. We compare trained Shortcut and Shortcut-CSL models for CIFAR10 by generating samples from random noise at $t = 0$ and measuring the misalignment u_θ between the two-step and it's base four-step trajectories at intermediate time steps $t = 0.5$ and $t = 1.0$. For a two-step trajectory, the value of u_θ at $t = 0.5$ and $t = 1.0$ correspond to immediate and future misalignments, respectively. As shown in Table 2, while both methods perform similarly at $t = 0.5$, Shortcut-CSL has a notably lower value of u_θ at $t = 1.0$, demonstrating its advantage in reducing future misalignments.

Table 2: Squared misalignment (u_θ^2) for CIFAR-10, averaged over 100 samples.

Method	$t = 0.5$	$t = 1.0$
ST	0.5×10^{-3}	2.5×10^{-3}
ST-CSL	0.5×10^{-3}	1.4×10^{-3}

6.2 Performance with Increasing Backbone Network Size

We evaluate whether the improvement of ST-CSL over ST is consistent across different backbone network sizes. We train both models using small (DiT-S-2, 74M parameters), medium (DiT-B-2, 296M parameters), and large (DiT-L-2, 1.2B parameters) diffusion transformer networks. The FID scores for models trained on CelebA and CIFAR10 images are shown in Figure 3. We observe that as the model size increases, the overall few-step generation quality of the shortcut model improves. Importantly, ST-CSL consistently outperforms ST across all model sizes and generation steps. This shows that the improvement of CSL over SL is robust across varying network sizes.

Table 3: Wall-clock time (in hours) for training (100 epochs) a Flow-Matching (FM), a shortcut (ST) and Ours (ST-CSL) methods for Diffusion Transformers of different sizes.

	FM	**ST**	**ST-CSL**
DiT-S-2 (74M)	–	5.6	6.2
DiT-B-2 (296M)	6.4	7.4	7.8
DiT-L-2 (1.2B)	–	11.4	12.2

Training Time Comparison: In Table 3, we compare the time taken to train the flow-matching (FM), shortcut(ST), and our methods. Training FM is efficient of all but it doesn't support few-step generation. Compared to ST, our method only consumes 7% more training time on average. Please refer to Appendix G for detailed time and memory consumption analysis.

Table 4: FID score comparison between shortcut and shortcut-CSL (ours) for different ratios of flow-matching to bootstrap targets ($B : K$). Our model consistently outperforms shortcut model across all ratios and generation steps on both CelebA-256 and CIFAR10 datasets.

		CelebA-256				**CIFAR10**			
B:K	Method	128-Step	Four-Step	Two-Step	One-Step	128-Step	Four-Step	Two-Step	One-Step
4:1	**ST**	**7.83**	9.36	12.56	20.46	7.37	9.15	11.79	19.80
	ST-CSL	7.88	**8.98**	**10.96**	**18.37**	**7.13**	**8.10**	**9.24**	**17.76**
2:1	**ST**	**7.73**	9.00	11.51	18.50	6.98	8.53	11.00	18.33
	ST-CSL	7.75	**8.58**	**10.09**	**16.37**	**6.95**	**7.98**	**9.08**	**16.51**
1:1	**ST**	8.01	9.00	11.14	16.51	**6.56**	8.17	10.54	16.43
	ST-CSL	**7.56**	**8.54**	**9.91**	**15.51**	6.67	**6.95**	**7.94**	**13.96**

6.3 Analysing the Effect of the Ratio $B : K$

We further evaluate whether the improvement of ST-CSL is consistent if we vary the ratio of flow-matching to bootstrap targets ($B : K$) while training. We report the FID scores with varying ratio in Table 4. Although increasing the percentage of bootstrap targets introduces additional computational cost, our results show that it significantly enhances the generation performance. Importantly, our ST-CSL consistently outperforms the ST baseline in few-step generation across all ratios. Moreover, we observe that at the $1 : 1$ ratio, our method achieves two-step performance within 2.0 points of the ST's 128-step performance on CelebA, and within 1.5 points on CIFAR-10.

6.4 Effect of R

We investigate the effect of increasing the number of terms in the CSL objective in Equation 14. Specifically, we experiment with $R = 1, 2, 4$, where larger values of R yield more accurate estimations of the CSL. Note that $R = 1$ corresponds to the SL loss used in the shortcut models. For all settings of R, the number of flow-matching and bootstrap targets per batch is kept constant. We observe that, compared to $R = 1$, $R = 2$ incurs approximately a 5% more training time, while $R = 4$ incurs about 30% more time. The FID scores for few-step generation under different values of R are reported in Table 5. The results demonstrate that increasing R consistently improves generation performance, highlighting the benefit of more accurate CSL estimation.

Table 5: FID-50K scores for one- and few-step generation for different values of R. Increasing the value of R results in more accurate estimation of the CSL loss and results in better sample quality.

	CelebA-256			**CIFAR10**		
R	Four-Step	Two-Step	One-Step	Four-Step	Two-Step	One-Step
1	9.00± 0.06	11.14± 0.03	16.51± 0.06	8.17± 0.10	10.54± 0.08	16.43± 0.06
2	8.54± 0.06	9.91± 0.05	15.51± 0.04	6.95± 0.07	7.94± 0.07	13.96± 0.10
4	**8.37± 0.03**	**9.38± 0.03**	**14.80± 0.09**	**6.66± 0.11**	**7.11± 0.05**	**13.10± 0.06**

7 Conclusion and Future Work

We formulate few-step generation as a controlled process, using the flow-matching model as its base process. This perspective provides the theoretical foundation for the self-consistency loss and motivates our proposed cumulative self-consistency loss. By training with this objective, we achieve significant improvement in one- and few-step generation performance over the baselines.

In this work, we consider the objective J without any intermediate state cost, i.e., $g(X_t^u, t) = 0$. An interesting direction for future work is to explore ways of incorporating this cost to influence the trajectory by penalizing undesirable intermediate states. Furthermore, the connection of our formulation to reinforcement learning offers opportunities to leverage techniques from the reinforcement learning literature to further enhance few-step generation quality.

REFERENCES

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

David Berthelot, Arnaud Autef, Jierui Lin, Dian Ang Yap, Shuangfei Zhai, Siyuan Hu, Daniel Zheng, Walter Talbott, and Eric Gu. Tract: Denoising diffusion models with transitive closure time-distillation. *arXiv preprint arXiv:2303.04248*, 2023.

Arthur E. Bryson and Yu-Chi Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. Taylor & Francis, New York, 1975.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *International Conference on Learning Representations (ICLR)*, 2025.

Floor Eijkelboom, Grigory Bartosh, Christian A. Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. In *Neural Information Processing Systems (NeurIPS)*, 2024.

Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning (ICML)*, 2024.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations (ICLR)*, 2025.

Sandesh Ghimire, Jinyang Liu, Armand Comas, Davin Hill, Aria Masoomi, Octavia Camps, and Jennifer Dy. Geometry of score based generative models. *arXiv preprint arXiv:2302.04411*, 2023.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Lingjie Liu, and Joshua M Susskind. Boot: Data-free distillation of denoising diffusion models with bootstrapping. In *ICML 2023 Workshop on Structured Probabilistic Inference and Generative Modeling*, 2023.

Agrim Gupta, Lijun Yu, Kihyuk Sohn, Xiuye Gu, Meera Hahn, Fei-Fei Li, Irfan Essa, Lu Jiang, and José Lezama. Photorealistic video generation with diffusion models. In *European Conference on Computer Vision(ECCV)*, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. In *International Conference on Machine Learning (ICML)*, 2023.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning (ICML)*, 2022.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations (ICLR)*, 2021.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. Technical Report.

Sangyun Lee, Zinan Lin, and Giulia Fanti. Improving the training of rectified flows. In *Neural Information Processing Systems (NeurIPS)*, 2024.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *Conference on Learning Representations (ICLR)*, 2023.

Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. AudioLDM: Text-to-audio generation with latent diffusion models. In *International Conference on Machine Learning (ICML)*, 2023a.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023b.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and qiang liu. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. In *International Conference on Learning Representations (ICLR)*, 2025.

Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

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.

Zhengxiong Luo, Dayou Chen, Yingya Zhang, Yan Huang, Liangsheng Wang, Yujun Shen, Deli Zhao, Jinren Zhou, and Tien-Ping Tan. Videofusion: Decomposed diffusion models for high-quality video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023b.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Hannes Risken. *The Fokker-Planck Equation: Methods of Solution and Applications*. Springer Series in Synergetics. Springer, 2nd edition, 1996. ISBN 978-3-540-61543-6.

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 (CVPR)*, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning (ICML)*, 2023.

Christopher Iliffe Sprague, Arne Elofsson, and Hossein Azizpour. Stable autonomous flow matching. *arXiv preprint arXiv:2402.05774*, 2024.

Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations (ICLR)*, 2023.

Luran Wang, Chaoran Cheng, Yizhen Liao, Yanru Qu, and Ge Liu. Training free guided flow-matching with optimal control. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 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 (CVPR)*, 2024.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. UniPC: A unified predictor-corrector framework for fast sampling of diffusion models. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Hongkai Zheng, Weili Nie, Arash Vahdat, Kamyar Azizzadenesheli, and Anima Anandkumar. Fast sampling of diffusion models via operator learning. In *International Conference on Machine Learning (ICML)*, 2023.

APPENDIX

A DERIVATION OF EQUATION 4

We have,

$$J(u_\theta; x_{t'}, t') = \int_{t'}^{1} \Big(f\big(u_\theta(X_t^u, t), t\big) + g(X_t^u, t) \Big)\, dt + h(X_1^u), \quad x_{t'} = X_{t'}^u \tag{15}$$

We can write the gradient of J as:

$$\begin{aligned} &\frac{d}{d\theta} \int_{t'}^{1} f(u_\theta(X_t^u, t), t) + g(X_t^u, t)\, dt + h(X_1^u) \\ &= \int_{t'}^{1} \frac{\partial}{\partial \theta} f(u_\theta(X_t^u, t), t) dt + \int_{t'}^{1} \frac{\partial [f(u_\theta(X_t^u, t), t) + g(X_t^u, t)]}{\partial X_t^u} \frac{\partial X_t^u}{\partial \theta}\, dt + \frac{\partial h(X_1^u)}{\partial \theta} \end{aligned} \tag{16}$$

Using the `stopgrad` notation as in Domingo-Enrich et al. (2025) for the last two terms

$$\int_{t'}^{1} \frac{\partial [f(u_\theta(X_t^u, t), t) + g(X_t^u, t)]}{\partial X_t^u} \frac{\partial X_t^u}{\partial \theta}\, dt + \frac{\partial h(X_1^u)}{\partial \theta} \tag{17}$$

$$= \frac{\partial}{\partial \theta} \left[\int_{t'}^{1} f(v(X_t^u, t), t) + g(X_t^u, t)\, dt + h(X_1^u) \right] \Bigg|_{v=\texttt{stopgrad}(u_\theta)} \tag{18}$$

Using the result from Domingo-Enrich et al. (2025). (Lemma 5),

$$\begin{aligned} &\frac{\partial}{\partial \theta} \int_{t'}^{1} f(v(X_t^u, t), t) + g(X_t^u, t)\, dt + h(X_1^u) \\ &= \int_{t'}^{1} \frac{\partial u_\theta(X_t^u, t)}{\partial \theta} \sigma(t) \nabla_{x_t} J(u_\theta; x_t, t)\, dt \end{aligned} \tag{19}$$

Plugging this into Equation 16,

$$\frac{dJ(u_\theta; x_{t'}, t')}{d\theta} = \int_{t'}^{1} \frac{\partial}{\partial \theta} f\big(u_\theta(X_t^u, t), t\big)\, dt + \int_{t'}^{1} \frac{\partial u_\theta(X_t^u, t)}{\partial \theta} \sigma(t) \nabla_{x_t} J(u_\theta; x_t, t) dt \tag{20}$$

B LATENT SPACE INTERPOLATABILITY

To analyze whether the one-step model learned by our approach captures an interpolatable latent space, we examine the model's outputs when fed with interpolated noise samples between two independently drawn Gaussian noise vectors. Specifically, given $x_0, x_0' \sim \mathcal{N}(0, I)$, we generate interpolated inputs using

$$x_\alpha = \sqrt{1-\alpha}\, x_0 + \sqrt{\alpha}\, x_0', \quad \alpha \in [0, 1].$$

We then apply one-step denoising to each interpolated sample using our shortcut-CSL model. The resulting images shown in Figure 4 exhibit smooth transitions across the interpolation path, suggesting that the model has an interpolatable latent space.

C SAMPLES FROM SHORTCUT-CSL

In Figure 5, we provide some samples from the proposed shortcut-CSL method for different generation budgets.

Figure 4: Image generated using the shortcut-CSL (ours) method with one-step denoising applied to a variance-preserving interpolation between two Gaussian noise samples. The leftmost and rightmost images correspond to independently drawn noise samples, while the intermediate images were produced from interpolated samples.

Figure 5: Image generated by the shortcut-CSL (ours) method by four-, two-, and one-step denoising. 128 steps denoising corresponds to the base flow-matching model.

D Hyperparameter Details

For the experiment in Table 1, we use the medium-scale diffusion transformer DiT-B-2 Peebles & Xie (2023) as the backbone network for both the shortcut baseline and our method. For CelebA-256, we downsample images from `3×256×256` to `4×32×32` using the VAE encoder from the Stable Diffusion framework Rombach et al. (2022), specifically the `sd-vae-ft-ema` variant. We then train the diffusion model in this compressed latent space. Detailed hyperparameters for our (ST-CSL) approach are provided in Table 6. Here, M denotes the total number of denoising steps in the base flow-matching model. The baseline shortcut method (ST) uses the same hyperparameter configuration, except with $R = 1$. The experiments were run on NVIDIA A100 and RTX A5000 GPUs and took about 24 hours to run.

Table 6: Hyperparameters used for training on CelebA-256 and CIFAR10.

Hyperparameter	Value
batch size	64
epochs	300 (CelebA-256), 500 (CIFAR10)
M	128
B	64
K	16
R	2
ema	0.9999
optimizer	AdamW
learning rate	0.0001
weight decay	0.1

E ALGORITHM

The algorithm for training shortcut models with the proposed CSL loss is detailed in Algorithm .

Algorithm 1 Training Shortcut Models with cumulative Self-Consistency Loss (Shortcut-CSL)

$K \leftarrow$ #bootstrap targets
$B \leftarrow$ #flow-matching targets
$R \leftarrow$ #terms in J_{CSL} estimation
while not converged **do**
  $x_0 \sim \mathcal{N}(0, I), \quad x_1 \sim D, \quad (d, t) \sim p(d, t)$
  $x_t \leftarrow (1-t)x_0 + tx_1$
  for B batch elements **do**
    $s_{\text{target}} \leftarrow x_1 - x_0$
    $d \leftarrow 0$
  end for
  for K/R batch elements **do**
    $t' \leftarrow t, x'_1 \leftarrow x_t, s_{\text{target}} = []$
    for R iterations **do**
      $s_1 \leftarrow s_\theta(x'_1, t', d)$
      $x'_2 \leftarrow x'_1 + s_1 d$
      $s_2 \leftarrow s_\theta(x'_2, t' + d, d)$
      APPEND$(s_{\text{target}}, \texttt{stopgrad}((s_1 + s_2)/2))$
      $x'_1 \leftarrow x'_1 + s_\theta(x'_2, t' + d, 2d).2d$
      $t' = t' + 2d$
    end for
  end for
  $t' \leftarrow t, x'_1 \leftarrow x_t$
  for $r = 1$ to R **do**
    $\theta \leftarrow \nabla_\theta ||s_\theta(x'_1, t, 2d) - s_{\text{target}}[r]||^2$
    $x'_1 = x'_1 + s_\theta(x'_1, t, 2d).2d$
    $t' = t' + 2d$

  end for
end while

F CONNECTION TO REINFORCEMENT LEARNING

Regarding the connection to reinforcement learning, we can formulate our method as a reinforcement problem as follows:

Starting with random noise at $t = 0$, an agent aims to transform it into a meaningful image in $(1/d)$-steps, taking a each step of size d going from $t = 0$ to $t = 1$. The agent is trained to take a direction, moving along which generates the best image (i.e., same as the direction taken by the base 128-step model in our case). For this, we define the reward as the negative of the CSL ($-J_{CSL}$), which we aim to maximize.

Doing it in the reinforcement learning way, $-J_{CSL}$ acts as our value function, and we can train a value network to estimate it. We can use the following temporal difference learning to learn a value network V_ϕ:

$$V_\phi(x_t, t, d) = -\|u_\theta(x_t, t, d)\|^2 + \gamma\, V_\phi(x_{t+d}, t + d, d) \tag{21}$$

where x_{t+d} is calculated using Equation 6. Here, $u_\theta(x_t, t, d)$ is the immediate misalignment where θ is the parameter of the few-step model, and V_ϕ represents the future misalignment from time $t + d$ onwards, predicted by the value network.

In reinforcement learning terms, the $(1/d)$-step model serves as an *actor*, and the value model serves as a *critic*, where the actor tries to maximize the estimated value. Instead of training a separate value network to estimate CSL, we opted for K-rollout as it offered a good balance of performance and efficiency. We leave further exploration with techniques from the reinforcement learning literature for future work.

G Empirical Time and Memory Consumption Comparison

Table 7: Wall-clock time (in hours) and GPU memory usage (in GB) for training a Flow-Matching (FM), a shortcut (ST) and Ours (ST-CSL) methods for Diffusion Transformers (parameter counts) of small, medium, and large sizes.

Model	Time (hours)			Memory (GB)		
	FM	**ST**	**ST-CSL**	**FM**	**ST**	**ST-CSL**
DiT-S-2 (74M)	–	5.6	6.2	–	21.35	21.35
DiT-B-2 (296M)	6.4	7.4	7.8	28.66	31.67	28.96
DiT-L-2 (1.2B)	–	11.4	12.2	–	45.47	38.52

Table 7 reports the wall-clock time (in hours) and GPU memory usage (in GB) for training Flow-Matching (FM), shortcut model (ST), and our method (ST-CSL) on the CelebA dataset for 100 epochs, using small, medium, and large Diffusion Transformer (DiT) models. All experiments were conducted on a single NVIDIA A100 GPU.

FM is the most efficient of all in time and memory, but does not support few-step generation.

Our method requires slightly more time than the shortcut model but uses less memory in larger models. This trade-off arises from how each method processes batches during training. The shortcut model processes the entire batch in a single step from the current state, which minimizes time but leads to high memory consumption. In contrast, our method uses a two-step process: it first computes an intermediate step using part of the batch, then computes the next step using the intermediate outputs (as described in lines 23–26 in Algorithm E). Splitting the batch this way reduces memory usage but requires additional time for the extra computation.

For two-stage methods like Progressive Distillation (PD) and Reflow, we first need to train the FM model and distill the knowledge to the student model. Reflow requires twice the time taken by FM. For PD, we train a student model that learns to generate samples in half as many steps as the teacher model. Therefore, a separate model needs to be trained for 128, 64, 32, 16, 8, 4, 2, and 1 step generation successively, consuming a significantly large amount of time.

H CONVERGENCE WITH TRAINING STEPS

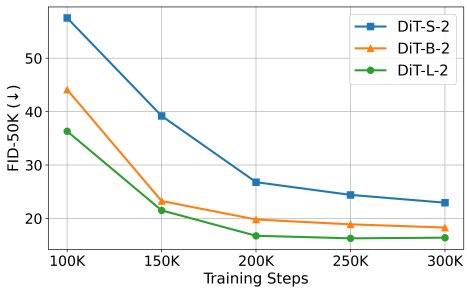

Figure 6: We evaluate the convergence of shortcut-CSL by tracking the FID score at regular training intervals across different backbone DiT network sizes on the CelebA-256 dataset. Our results show that performance consistently improves with increasing model size at every stage of training.

I LLM USAGE

LLM is lightly used to polish writing in the paper. The used prompt is `Please polish the following sentence/sentences`