

---

# Improved Training Technique for Shortcut Models

---

Anh Nguyen\* Viet Nguyen\*† Duc Vu Trung Dao Chi Tran Toan Tran Anh Tran

Qualcomm AI Research‡

## Abstract

Shortcut models represent a promising, non-adversarial paradigm for generative modeling, uniquely supporting one-step, few-step, and multi-step sampling from a single trained network. However, their widespread adoption has been stymied by critical performance bottlenecks. This paper tackles the five core issues that held shortcut models back: (1) the hidden **flaw of compounding guidance**, which we are the first to formalize, causing severe image artifacts; (2) **inflexible fixed guidance** that restricts inference-time control; (3) a pervasive **frequency bias** driven by a reliance on low-level distances in the direct domain, which biases reconstructions toward low frequencies; (4) **divergent self-consistency** arising from a conflict with EMA training; and (5) **curvy flow trajectories** that impede convergence. To address these challenges, we introduce iSM, a unified training framework that systematically resolves each limitation. Our framework is built on four key improvements: *Intrinsic Guidance* provides explicit, dynamic control over guidance strength, resolving both compounding guidance and inflexibility. A *Multi-Level Wavelet Loss* mitigates frequency bias to restore high-frequency details. *Scaling Optimal Transport (sOT)* reduces training variance and learns straighter, more stable generative paths. Finally, a *Twin EMA* strategy reconciles training stability with self-consistency. Extensive experiments on ImageNet  $256 \times 256$  demonstrate that our approach yields substantial FID improvements over baseline shortcut models across one-step, few-step, and multi-step generation, making shortcut models a viable and competitive class of generative models.

## 1 Introduction

Recent generative diffusion models based on flow matching have achieved remarkable success in synthesizing high-fidelity data across various domains [14, 32, 58]. While demonstrably successful, a significant bottleneck persists: generating samples requires integrating the learned vector field over many discrete timesteps, resulting in high computational costs that limit deployment in resource-constrained environments and latency-sensitive applications. Accelerating sampling without sacrificing quality has thus become a critical research area.

Various acceleration techniques have emerged, including timestep distillation [42, 11, 66, 65, 8], advanced numerical solvers [40], lightweight architectures [7, 61, 64], and single-stage training procedures [54, 69, 18]. Shortcut models (SM) [18] offer a particularly elegant approach, using a generator conditioned on both noise level  $t$  and desired step size  $d$  with an additional self-consistency loss. This enables the prediction of multiple timesteps ahead in a single forward pass, supporting variable sampling budgets at inference using the same network. Despite this promising method, widespread adoption has been hindered by critical performance bottlenecks. This paper tackles the

---

\*Equal contribution

†Work done while at Qualcomm.

‡Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

five core issues that held shortcut models back: *the hidden flaw of compounding guidance, inflexible fixed guidance, frequency bias, divergent self-consistency, and curvy flow trajectories*.

The first two issues stem from a flawed integration of Classifier-Free Guidance (CFG). Not only are users locked into a fixed guidance scale at training time—sacrificing control over the diversity-fidelity trade-off—but this design also conceals a deeper flaw. We are the first to formalize that this fixed guidance *compounds exponentially* across the implicit sub-steps of a large generation step, causing severe image artifacts. The other three issues further degrade sample quality: a reliance on pixel-wise losses creates a *frequency bias* toward blurry images; random noise-data pairings create unstable, *high-curvature generative paths*; and a *temporal lag in the EMA target* prevents the model from learning true self-consistency for large jumps.

To address these challenges, we introduce the **Improved Shortcut Model (iSM)**, a unified training framework that systematically resolves each limitation through four key components.

1. **Intrinsic Guidance:** We resolve both guidance-related flaws by making the guidance scale an explicit input to the model. This provides *dynamic, inference-time control*, works out-of-the-box for one-step generation, and reduces inference time by  $\sim 50\%$  compared to standard CFG, all while correcting the compounding effect.
2. **Multi-Level Wavelet Loss:** We replace standard pixel-wise objectives with a *frequency-aware loss* that forces the model to reconstruct high-frequency details, mitigating frequency bias.
3. **Scaling Optimal Transport (sOT):** We straighten generative trajectories by periodically pooling samples from several mini-batches to compute a large-scale transport plan. This *decouples the OT batch size from the training batch size*, yielding more stable paths without a heavy computational cost.
4. **Twin EMA Strategy:** We eliminate target lag by *maintaining two EMA networks*—a fast-decay one to generate fresh, up-to-date consistency targets and a slow-decay one to ensure stable, high-quality inference.

With these improvements, iSM achieves FID scores of 5.27 and 2.05 on ImageNet  $256 \times 256$  with just one and four sampling steps, respectively. Our work *closes the performance gap* with leading generative models and establishes improved shortcut models as a *flexible, efficient, and highly competitive* modeling paradigm.

## 2 Preliminaries

**Flow Matching (FM)** [37, 38] offers an elegant framework for generative modeling of data distributions  $p_{\text{data}}(\mathbf{x})$ . At its core, FM defines  $\mathbf{x}_t = (1 - t)\mathbf{x}_0 + t\mathbf{x}_1$  as a linear interpolation between a noise sample  $\mathbf{x}_0$  drawn from a standard normal distribution  $\mathcal{N}(0, \mathbf{I})$ , denoted  $\mathcal{N}$ , and a data sample  $\mathbf{x}_1$  drawn from the data distribution  $D$ . Here,  $t \in [0, 1]$  represents the timestep, parameterizing the interpolation from noise ( $t = 0$ ) to data ( $t = 1$ ). It then trains a velocity model  $\bar{\mathbf{v}}_\theta(\mathbf{x}_t, t, c)$  to match the ground-truth velocity field  $\mathbf{v} = \mathbf{x}_1 - \mathbf{x}_0$  by minimizing the following velocity loss:

$$\mathcal{L}_{\text{velocity}}(\theta) = \mathbb{E}_{\substack{\mathbf{x}_0 \sim \mathcal{N}, (\mathbf{x}_1, c) \sim D \\ t \sim p(t)}} [\|\bar{\mathbf{v}}_\theta(\mathbf{x}_t, t, c) - \mathbf{v}\|^2]. \quad (1)$$

Here,  $\mathbf{x}_t$  is the sample interpolated at time  $t$ , and  $c$  represents any conditioning information, such as text or class labels, associated with the data sample  $\mathbf{x}_1$ . Sampling from such a model typically involves discretizing the learned ODE using numerical methods like Euler integration, often requiring dozens or hundreds of steps. Naively taking large steps with  $\bar{\mathbf{v}}_\theta$  leads to significant discretization errors, as the predicted velocity points towards an average of potential target data points, causing mode collapse.

**Shortcut Models (SM).** To address this limitation and enable efficient few-step and one-step generation, [18] proposes training a shortcut model that conditions the neural network  $\mathbf{s}_\theta(\mathbf{x}_t, t, c, d)$  not only on the current timestep  $t$  and the condition  $c$ , but also on a desired step size  $d$ . This allows the model to predict the normalized displacement needed to directly reach the next point  $\mathbf{x}'_{t+d}$  at time  $t + d$ , thereby bypassing intermediate steps of the probability flow ODE.

The training approach includes *flow-matching* for infinitesimal steps and *self-consistency* for larger steps. At infinitesimal step sizes ( $d \approx 0$ ), shortcut models use the flow-matching objective, regressing

the model  $s_\theta(\mathbf{x}_t, t, c, d = 0)$  to predict the empirical velocity  $\mathbf{s}_{\text{velocity}}$ , similar to traditional flow-matching models. For larger step sizes ( $d > 0$ ), shortcut models leverage a self-consistency property where one large shortcut step is equivalent to two consecutive smaller shortcut steps of half the size. This property allows the model to learn efficient large-step transitions by recursively breaking them into smaller steps. To implement this self-consistency objective for training, they select a base number of steps  $N$  ( $N = 128$  in practice), which defines the smallest time unit for the ODE approximation. This results in  $\log_2(N) + 1 = 8$  distinct shortcut lengths available during training, specifically  $d \in \{1/128, 1/64, \dots, 1/2, 1\}$ , over which the self-consistency loss is applied. When  $d$  is at the smallest value (e.g.,  $1/128$ ), they instead query the model at  $d = 0$ . CFG [24], widely used to improve conditional sample quality, is integrated within SM to form guided targets  $\mathbf{g}_\theta^w$  based on the model output  $s_\theta$ :

$$\mathbf{g}_\theta^w(\mathbf{x}_t, t, c, d) = s_\theta(\mathbf{x}_t, t, c, d) + w \cdot (s_\theta(\mathbf{x}_t, t, c, d) - s_\theta(\mathbf{x}_t, t, \emptyset, d)), \quad (2)$$

where  $\emptyset$  denotes the null condition. Using the guidance-scaled output  $\mathbf{g}_\theta^w$ , the self-consistency target  $\mathbf{s}_{\text{consistency}}$  is constructed by simulating two consecutive steps of size  $d$ :

$$\mathbf{s}_{\text{consistency}} = \mathbf{g}_\theta^w(\mathbf{x}_t, t, c, d)/2 + \mathbf{g}_\theta^w(\mathbf{x}'_{t+d}, t + d, c, d)/2, \quad (3)$$

where  $\mathbf{x}'_{t+d} = \mathbf{x}_t + \mathbf{g}_\theta^w(\mathbf{x}_t, t, c, d)d$ . These components then form a unified loss function:

$$\mathcal{L}^S(\theta) = \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}, (\mathbf{x}_1, c) \sim D} \left[ \underbrace{\|\mathbf{s}_\theta(\mathbf{x}_t, t, c, d = 0) - \mathbf{s}_{\text{velocity}}\|^2}_{\text{Flow-Matching}} + \underbrace{\|\mathbf{s}_\theta(\mathbf{x}_t, t, c, 2d) - \mathbf{s}_{\text{consistency}}\|^2}_{\text{Self-Consistency}} \right]. \quad (4)$$

This approach allows shortcut models to achieve effective multi-step, few-step, and single-step generation using a single trained network.

### 3 Improved Guidance Sampling for Shortcut Models

Shortcut models [18] represent a significant step toward efficient one-step, few-step, and many-step generation. However, their integration of CFG introduces unique challenges compared to standard diffusion methods. This section revisits the shortcut model formulation to identify and address two critical limitations: (1) *inflexibility due to fixed guidance scales*, (2) an *overlooked flaw of accumulation* when applying CFG.

#### 3.1 Problem Statement

**Inflexibility of Fixed Guidance.** The original shortcut model framework requires using a fixed guidance scale  $w$  during training. This approach is restrictive for two main reasons. First, selecting an optimal  $w$  is a difficult hyperparameter tuning problem that depends on the model's final behavior and the intended number of inference steps. Second, and more critically, it removes a key control for balancing the fidelity-diversity trade-off at inference time, substantially reducing the model's *flexibility in production settings*.

**Flaw of Accumulated Guidance.** A critical, previously overlooked issue in shortcut models is the accumulation of guidance effects. We are the first to identify and formalize this critical flaw, which stems from the recursive application of a fixed guidance scale. When a shortcut model generates a sample in a single large step, it implicitly combines the effects of many smaller, guided steps. This *compounding characteristic* leads to an *unintended amplification* of the guidance signal.

To formalize this, we consider a shortcut model  $s_\theta$  and its corresponding classifier-free guided output  $\mathbf{g}_\theta^w$  with guidance scale  $w$ . To analyze the generation process, we define a sequence of intermediate points  $\{\mathbf{x}'_{\frac{i}{N}}\}_{i=0}^N$ . This sequence is constructed by recursively applying  $N$  consecutive shortcut steps, each with the smallest size  $d = 1/N$ . The process begins from the initial noise  $\mathbf{x}_0$  at  $t = 0$ , conditioned on  $c$ . The starting point of the sequence is thus defined as  $\mathbf{x}'_0 = \mathbf{x}_0$ , and subsequent intermediate points are generated as follow:

$$\mathbf{x}'_{\frac{i+1}{N}} = \mathbf{x}'_{\frac{i}{N}} + s_\theta\left(\mathbf{x}'_{\frac{i}{N}}, \frac{i}{N}, c, d\right)d, \quad \text{for } i = 0, \dots, N-1. \quad (5)$$

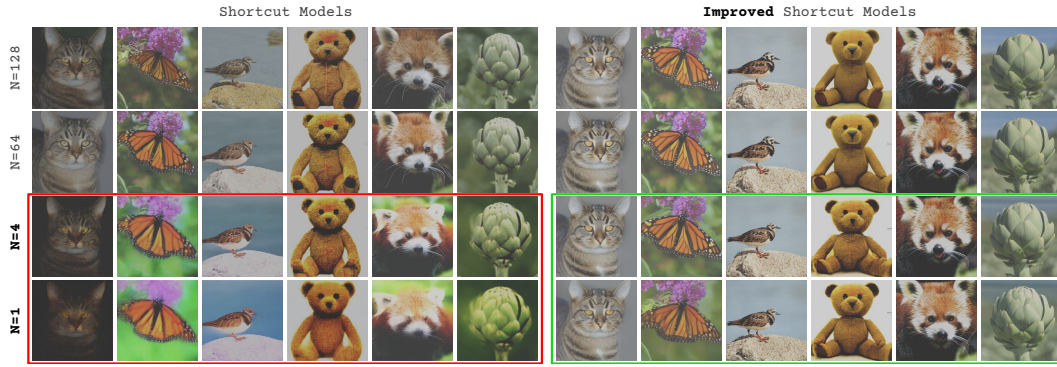


Figure 1: Visualizing the *accumulated guidance* problem in standard shortcut models (left) versus our integrated-guidance approach (right). Top row shows multi-step generation; bottom row shows one-step generation. In the original shortcut model, the fixed guidance strength learned at training time effectively compounds across the many implicit sub-steps, leading to oversaturated colors and blurriness. By conditioning the network on the guidance scale and training it to apply guidance explicitly at each step, our method produces consistent, artifact-free images even in the extreme one-step and few-step regimes.

**Proposition 1.** *The model’s prediction for a single large shortcut step of size  $Nd = 1$  approximately equals the average of the guided displacements corresponding to the  $N$  smallest steps, but with an exponentially compounded guidance scale:*

$$s_{\theta}(x_0, 0, c, Nd) \approx \frac{1}{N} \sum_{i=0}^{N-1} g_{\theta}^{w^{\log_2(N)}} \left( x'_{\frac{i}{N}}, \frac{i}{N}, c, d \right). \quad (6)$$

*Proof.* See Appendix A. □

The original shortcut model [18] defines a fixed CFG scale  $w = 1.5$ , exclusively applied when constructing self-consistency targets in Eq. (3). Consequently, the one-step generation  $s_{\theta}(x_0, 0, c, 1)$  implicitly aggregates the cumulative effect of  $N$  intermediate steps. As formalized in Proposition 1, the guidance scale at each of these implicit intermediate steps is not  $w$  but a much higher value  $w' = w^{\log_2(N)}$ . For  $N = 128$  base steps and  $w = 1.5$ , this calculation yields an extremely high intermediate guidance scale for single large shortcut step cases ( $w' = 1.5^{\log_2(128)} \approx 17$ ). Such high implicit guidance causes artifacts like *over-saturation*, particularly in one-step and few-step generations (see Fig. 1).

### 3.2 Intrinsic Guidance Training for Shortcut Models

To address these limitations of shortcut models [18], we introduce an *Intrinsic Guidance* training framework. In contrast to CFG implementations that operate between model outputs, as commonly practiced in multi-step diffusion models, we integrate the guidance mechanism into the *network’s internal space*. The core idea is to make the guidance scale  $w$  an *explicit input* to the model and to train the model to produce *guided outputs directly* across a range of scales. This framework enables flexible CFG for few-step and many-step generation and, importantly, provides a principled way to apply CFG for one-step generation while resolving the accumulated guidance issue.

**Model Parameterization.** We define the velocity field generator as a neural network  $s_{\theta}$ . Unlike previous shortcut models [18],  $s_{\theta}$  receives the CFG scale  $w \geq 0$  as an additional input, alongside the current sample  $x_t$ , time  $t$ , condition  $c$ , and the target step size  $d$ . The network is thus trained to directly output the CFG-modulated velocity  $s_{\theta}(x_t, t, c, d, w)$ , allowing guided sample generation with a single network evaluation per step. For implementation details, please refer to Appendix C.

**Flow Matching Objective.** This objective [37, 38] guides the model to predict the conditional and unconditional velocity for infinitesimal steps ( $d = 0$ ) without guidance ( $w = 0$ ), establishing the



base vector fields upon which guidance will be built:

$$\mathcal{L}_{\text{velocity}}(\theta) := \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}, (\mathbf{x}_1, c) \sim D} \left[ \left\| \mathbf{s}_\theta(\mathbf{x}_t, t, c, d=0, w=0) - \mathbf{s}_{\text{velocity}} \right\|^2 \right], \quad (7)$$

where  $\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1$  and  $\mathbf{s}_{\text{velocity}} = \mathbf{x}_1 - \mathbf{x}_0$ .

When sampling the condition  $c$  for training, we randomly include the null condition  $\emptyset$  via stochastic dropout, following the standard scheme [41].

**Intrinsic Guidance Objective.** This objective guides the model to directly produce the CFG-scaled output for infinitesimal steps ( $d=0$ ) when being conditioned on non-zero guidance strengths ( $w>0$ ). We begin by recalling the standard formulation of CFG with  $w>0$ :

$$\mathbf{s}_\theta(\cdot, c, d=0, w) \approx \mathbf{s}_\theta(\cdot, c, d=0, w=0) + w \cdot \underbrace{(\mathbf{s}_\theta(\cdot, c, d=0, w=0) - \mathbf{s}_\theta(\cdot, \emptyset, d=0, w=0))}_{\mathbf{s}_{\text{guidance}}}, \quad (8)$$

where  $\mathbf{s}_{\text{guidance}}$  is the estimated guidance direction using the model's base prediction ( $w=0$ ).

We substitute the unguided velocity  $\mathbf{s}_\theta(\cdot, c, d=0, w=0)$  in Eq. (7) based on the approximation of  $\mathbf{s}_\theta(\cdot, c, d=0, w=0)$  in Eq. (8):

$$\begin{aligned} \mathbf{s}_\theta(\cdot, c, d=0, w=0) &\approx \mathbf{s}_\theta(\cdot, c, d=0, w) - w \cdot \mathbf{s}_{\text{guidance}}, \\ \left\| \mathbf{s}_\theta(\cdot, c, d=0, w=0) - \mathbf{s}_{\text{velocity}} \right\|^2 &\approx \left\| (\mathbf{s}_\theta(\cdot, c, d=0, w) - w \cdot \mathbf{s}_{\text{guidance}}) - \mathbf{s}_{\text{velocity}} \right\|^2 \\ &= \left\| \mathbf{s}_\theta(\cdot, c, d=0, w) - (\mathbf{s}_{\text{velocity}} + w \cdot \mathbf{s}_{\text{guidance}}) \right\|^2. \end{aligned}$$

Integrating this derived target into Eq. (7), we define a new training objective called the intrinsic guidance loss  $\mathcal{L}_{\text{guidance}}$ . Crucially, we apply the stop-gradient operator  $\text{sg}(\cdot)$  to the guidance direction  $\mathbf{s}_{\text{guidance}}$  within this loss. This *isolates the optimization* to focus solely on how the network learns to scale its output in response to  $w>0$ , preventing interference with the optimization of the foundational  $w=0$  predictions in  $\mathcal{L}_{\text{velocity}}$  and improving training stability.

$$\mathcal{L}_{\text{guidance}}(\theta) := \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}, (\mathbf{x}_1, c) \sim D} \left[ \left\| \mathbf{s}_\theta(\mathbf{x}_t, t, c, d=0, w) - (\mathbf{s}_{\text{velocity}} + w \cdot \text{sg}(\mathbf{s}_{\text{guidance}})) \right\|^2 \right], \quad (9)$$

where guidance scale  $w>0$  is drawn from a distribution  $p(w)$ . This loss trains the network to learn how CFG works for infinitesimal steps ( $d=0$ ).

**Guided Self-Consistency Objective.** This objective generalizes the self-consistency principle from [18] to operate with arbitrary step sizes ( $d>0$ ) and any guidance scale ( $w \geq 0$ ). The objective maintains the foundational properties of shortcut models, where a *single, large guided shortcut step* yields an output consistent with the composition of *two smaller, consecutive guided steps*.

$$\begin{aligned} \mathcal{L}_{\text{consistency}}(\theta) &:= \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}, (\mathbf{x}_1, c) \sim D} \left[ \left\| \mathbf{s}_\theta(\mathbf{x}_t, t, c, 2d, w) - \mathbf{s}_{\text{consistency}} \right\|^2 \right], \\ \text{where } \mathbf{s}_{\text{consistency}} &:= \mathbf{s}_{\theta^-}(\mathbf{x}_t, t, c, d, w)/2 + \mathbf{s}_{\theta^-}(\mathbf{x}'_{t+d}, t, c, d, w)/2 \\ \text{and } \mathbf{x}'_{t+d} &= \mathbf{x}_t + \mathbf{s}_\theta(\mathbf{x}_t, t, c, d, w)d, \end{aligned} \quad (10)$$

where  $\theta^-$  is the EMA target network. The stop-gradient operator  $\text{sg}(\cdot)$  is applied to the entire consistency target to stabilize training, following standard practice for self-consistency objectives.

**Final Training Objective.** The final objective is a weighted sum of the individual components as follows:

$$\mathcal{L}_{\text{total}}(\theta) = \alpha \mathcal{L}_{\text{velocity}}(\theta) + \beta \mathcal{L}_{\text{guidance}}(\theta) + \gamma \mathcal{L}_{\text{consistency}}(\theta), \quad (11)$$

where  $\alpha, \beta, \gamma > 0$  are hyperparameters. For simplicity, we set  $\alpha = \beta = \gamma = 1$  in our experiments.

### 3.3 Interval Guidance in Training

Findings presented in [31] suggest that during the inference of diffusion models, strong classifier-free guidance can be *detrimental during the early stages* of the reverse process, corresponding to high noise levels (time  $t$  approaching 0 under the convention where  $t=0$  is pure noise).

At high noise levels, the signal-to-noise ratio is low. Hence, the unconditional prediction  $s_\theta(\mathbf{x}_t, t, \emptyset, d = 0, w = 0)$  points towards the average image in the dataset  $D$ , while the conditional prediction  $s_\theta(\mathbf{x}_t, t, c, d = 0, w = 0)$  points vaguely towards the average image for that condition. Consequently, the guidance vector  $\mathbf{s}_{\text{guidance}}$  points roughly from the global data mean towards the conditional data mean. Applying strong extrapolation ( $w > 0$ ) along this coarse direction to a high-entropy state  $\mathbf{x}_t$  can cause the sample diversity to *collapse prematurely* towards a few dominant modes.

To mitigate this issue, we incorporate this insight into the training objective  $\mathcal{L}_{\text{guidance}}$  in Eq. (9) by applying guidance only within a defined time interval  $[t_{\text{interval}}, 1)$ . In particular, we modify the guidance scale  $w$  used within the Eq. (9) to be time-dependent:

$$w(t) = \begin{cases} w, & \text{if } t \in [t_{\text{interval}}, 1) \\ 0, & \text{otherwise,} \end{cases} \quad (12)$$

where  $w$  is the sampled guidance scale. This strategy trains the model to apply zero guidance at high noise levels, preserving diversity in the critical early phase of generation. We find that  $t_{\text{interval}} = 0.3$  works well empirically, and this threshold is represented by the red dashed line in Figs. 2b and 2c. Notably, in the high-noise regime where  $t < 0.3$  (i.e., before guidance is applied), both flow-matching and self-consistency losses exhibit lower loss pattern.

## 4 Multi-Level Wavelet Function Against Frequency Bias

Deep neural networks often prioritize learning *low-frequency information over high-frequency details*, a phenomenon known as frequency bias [47, 50, 3, 62, 63]. This bias is detrimental to generative models [10, 17, 29, 51], particularly shortcut models [18] optimized for few-step generation with low-level direct domain losses, such as the  $\ell_2$  loss. These models typically capture global structure but lack fine-grained, high-frequency details, resulting in blurred textures. The low-level  $\ell_2$  loss contributes to this, as it is often dominated by errors in low-frequency components [68].

To mitigate this frequency bias, we introduce optimizing the reconstruction objective in the *wavelet domain*. Specifically, discrete wavelet transform (DWT) is employed to decompose both the prediction and the target (Eqs. (7), (9) and (10)) into their respective wavelet coefficients, yielding multi-band representations, which introduce a *frequency-aware error signal*. By isolating reconstruction errors at different frequency levels, the optimization process is better guided to preserve high-frequency details that are often neglected in standard losses. Consequently, the model learns to generate outputs with significantly improved perceptual quality, capturing both the global structure and fine-grained details.

To further refine supervision across the frequency spectrum, we extend this to a multi-level DWT decomposition. This process involves recursively decomposing the wavelet sub-bands from the previous level. Our experiments consistently show that using a deeper multi-level decomposition, with five levels ( $L = 5$ ) in our experiments, substantially improves the synthesis of high-frequency content compared to a single-level DWT loss or standard pixel-wise objectives. Pseudocode is provided in Appendix B.

## 5 Scaling Optimal Transport for Better Trajectories

Shortcut models, similar to standard flow matching, often encounter considerable loss variance during training. This instability, as highlighted in prior works [46, 33], primarily stems from the *conventional random pairing* of initial noise samples  $\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I})$  with target images ( $\mathbf{x}_1 \sim D$ ). Such random coupling frequently results in *intersecting forward trajectories* ( $\mathbf{x}_t$ ), where similar noise inputs might be mapped to vastly different data targets. Consequently, the learned generative ODE exhibits *high-curvature reverse paths*, posing a significant challenge for shortcut models. Optimal Transport (OT) within mini-batches can mitigate this by re-assigning noise-image pairs to minimize a transport cost, typically  $L_2$  distance, yielding smoother trajectories [46, 33].

Although mini-batch OT helps, we recognize that the efficacy of such mini-batch OT is fundamentally linked to the number of samples considered. Theoretical and empirical findings [13] show that OT better approximates the true transport plan as the number of samples  $n$  increases. However, scaling

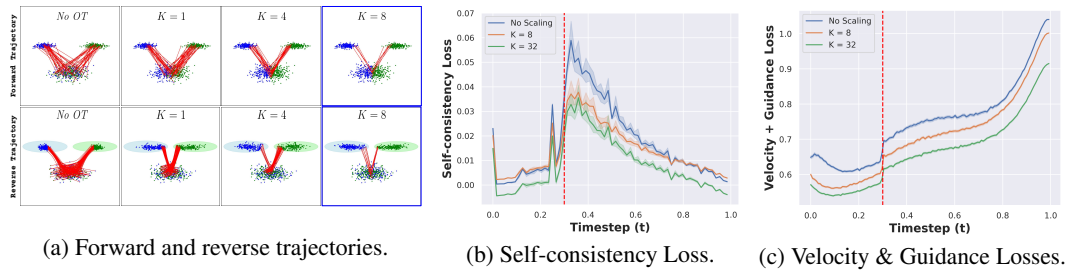


Figure 2: **Efficacy of Scaling Optimal Transport (sOT) Matching** in improving shortcut model training, demonstrated by varying the OT scaling factor  $K$ . **(a)** On a bimodal target, forward trajectories (top row) without OT exhibit frequent intersections (red), compelling the reverse generative process (bottom row) to follow high-curvature paths that initially average the target modes (blue, green). Our sOT approach, by increasing the effective OT scaling  $K$ , progressively disentangles these forward couplings, yielding substantially straighter reverse trajectories. **(b, c)** The benefits of this trajectory straightening are reflected in the training losses: both self-consistency (b) and velocity & guidance (c) losses are consistently lower and more stable with larger  $K$ , especially after the interval guidance point  $t_{\text{interval}} = 0.3$  (left-side of the red dashed line). **Note for (a):** To avoid overly dense trajectory, only intersecting forward paths and sampling trajectories with high curvature are highlighted. Zoom for detail.

the mini-batch size  $M$  to achieve this is often limited by GPU memory constraints during training. To tackle this scalability challenge, we introduce Scaling Optimal Transport (sOT), a strategy that decouples the effective batch size for OT computation from the batch size used for model training. Our method allows us to use a large number of samples for OT without increasing the memory needed for model training. Specifically, for every  $K$  training batches of size  $M$ , we aggregate all noise and image samples into a single set of size  $K \times M$  and compute one OT plan over it. The resulting  $K \times M$  matched pairs are then split back into  $K$  mini-batches of size  $M$  and used for the next  $K$  standard training steps. This strategy allows us to benefit from an effective OT batch size of  $K \times M$  while maintaining a computationally tractable batch size  $M$  for model training. The computational overhead from sOT is modest, adding approximately 4% to the total training time in our experiments.

Fig. 2a demonstrates sOT’s effectiveness on a 2D bimodal dataset. Random pairing (leftmost column) creates overlapping forward trajectories, forcing the reverse process to follow high-curvature paths that initially move toward an average direction before bending toward target modes. As the effective OT batch size  $K$  increases (columns 2-4), sOT produces progressively more disentangled forward couplings with fewer intersections, resulting in straighter generative trajectories.

Figs. 2b and 2c further illustrate the benefits of sOT by showing a consistent decrease in self-consistency and flow-matching losses, as the sOT scaling factor  $K$  increases, with these loss statistics computed over 1.2M ImageNet samples. Regarding self-consistency loss (Fig. 2b), sOT straightens the learned generative paths (as seen in Fig. 2a). On these smoother trajectories, a single large shortcut step more accurately approximates the outcome of two composed smaller steps, thus reducing the discrepancy and lowering the loss. This indicates that the model learns more reliable long-range predictions. For flow-matching and guidance loss (Fig. 2c), sOT provides more structured noise-data pairings. This results in a less conflicting target velocity field for the model to learn, as similar noise inputs are mapped to more consistently related data targets. This simplified regression task allows the model to learn a more coherent target, thereby reducing the error.

## 6 Twin EMA Improves Self-Consistency Property

Self-consistency objectives seek to align large-step generations with sequences of smaller steps. Specifically, they train an online network  $\theta$  to produce outputs from a  $2d$ -sized step that match the consistency target  $s_{\text{consistency}}$ , which is obtained by applying two consecutive  $d$ -sized steps. Instead of using the online network  $\theta$  to compute  $s_{\text{consistency}}$ , the original paper suggests using a target network  $\theta^-$ , typically implemented as an EMA of  $\theta$ . When utilized for generating self-consistency targets, EMA parameters provide crucial stability that dampens the oscillatory behaviors that would otherwise propagate and amplify across different timesteps.

Despite its essential stabilizing role, conventional EMA implementations introduce a tension in self-consistency training. When the target network  $\theta^-$  is maintained with a standard slow decay rate, it inevitably represents a historical state of the online network rather than its target distribution. This temporal lag means that self-consistency targets  $s_{\text{consistency}}$  are derived from outdated networks.

Consequently, the online network  $\theta$  faces a *conflicting objective*: it must simultaneously optimize for its current trajectory during many-step generation (i.e., flow matching loss) while aligning with targets generated from its historical states during few-step (i.e., self-consistency loss). As training progresses, this misalignment forces the model to learn and balance multiple, potentially contradictory generative mappings rather than converging toward a unified process. The slower the EMA decay rate, the more pronounced this distributional divergence becomes.

To resolve this tension while preserving the essential stabilizing benefits of EMA, we introduce a Twin EMA approach that maintains *two distinct parameter sets*: (1) **Inference Parameters** ( $\theta_{\text{infer}}^-$ ): Updated with a conventional slow decay rate and used exclusively during inference, ensuring high-quality sample generation benefits from long-term parameter averaging. (2) **Target Parameters** ( $\theta_{\text{target}}^-$ ): Updated with a significantly faster decay rate to maintain close proximity to the online network’s current state while still providing stabilization. This *decoupling addresses the fundamental conflict* between stability and recency requirements. By generating self-consistency targets from a near-contemporary version of itself, the online network more effectively enforces consistency across different timestep discretizations. Meanwhile, the separate maintenance of slow-decay parameters for inference preserves the stability crucial for high-quality sample generation.

## 7 Experiment

We combine all the improvements detailed from Sections 3 to 6 to train our improved shortcut models. We extensively benchmark our models on ImageNet  $256 \times 256$  against various models across multiple training paradigms. We evaluate sample quality using FID-50K [20]. For all experiments, our models use the XL/2 variant of the SiT architecture [41]. We report evaluations against other models at *800K iterations*, while component-wise analyses are conducted at *250K iterations*. We summarize our main quantitative benchmarks in Table 1, provide a broader evaluation using additional metrics in Table 2, present a detailed component-wise analysis in Table 3, and demonstrate the scalability and generality of our framework in Table 4.

### 7.1 Quantitative result and comparison

Our model, iSM, achieves strong FID scores across various NFE settings, demonstrating high sample quality and inference efficiency. With 8 inference steps, iSM *surpasses the 10-step VAR* [56] of similar model size and performs *comparably to VAR’s much larger 2B variant*. A key advantage of iSM is its inherent support for *flexible NFE counts*—from single-step to many-step generation—a feature absent in models like VAR, with FID scores significantly improving as NFE increases from 1 to 8. Furthermore, iSM performs favorably against other recent variable-step models, including the original shortcut model [18] and IMM [69]. For instance, in single-step generation, iSM attains an

Table 1: **Quantitative results** on class-conditional ImageNet  $256 \times 256$  across different model types.

| Model                              | FID-50k ( $\downarrow$ ) | NFE ( $\downarrow$ ) | #Params |
|------------------------------------|--------------------------|----------------------|---------|
| <b>GAN</b>                         |                          |                      |         |
| BigGAN-deep [5]                    | 4.06                     | 1                    | 112M    |
| GigaGAN [26]                       | 3.45                     | 1                    | 569M    |
| StyleGAN-XL [28]                   | 2.30                     | 1                    | 166M    |
| <b>Masked &amp; AR</b>             |                          |                      |         |
| VQGAN [16]                         | 26.52                    | 1024                 | 227M    |
| MaskGIT [6]                        | 6.18                     | 8                    | 227M    |
| VAR- <i>d</i> 20 [56]              | 2.57                     | 10                   | 600M    |
| VAR- <i>d</i> 30 [56]              | 1.92                     | 10                   | 2B      |
| MAR [35]                           | 1.98                     | 100                  | 400M    |
| <b>Diffusion &amp; Flow Models</b> |                          |                      |         |
| ADM [12]                           | 10.94                    | 250                  | 554M    |
| CDM [23]                           | 4.88                     | 8100                 | -       |
| SimDiff [25]                       | 2.77                     | 512                  | 2B      |
| LDM-4-G [48]                       | 3.60                     | 250                  | 400M    |
| U-DiT-L [57]                       | 3.37                     | 250                  | 916M    |
| U-ViT-H [2]                        | 2.29                     | 50                   | 501M    |
| DiT-XL/2 [44]                      | 2.27                     | 250                  | 675M    |
| SiT-XL/2 [41]                      | 2.15                     | 250                  | 675M    |
| REPA-XL/2 [67]                     | 1.42                     | 250                  | 675M    |
| FlowDCN [60]                       | 2.00                     | 250                  | 618M    |
| <b>One-to-Many Step Models</b>     |                          |                      |         |
| iCT [54]                           | 34.24                    | 1                    | 675M    |
|                                    | 20.3                     | 2                    | 675M    |
| SM [18]                            | 10.60                    | 1                    | 675M    |
|                                    | 7.80                     | 4                    | 675M    |
|                                    | 3.80                     | 128                  | 675M    |
| IMM [69]                           | 7.77                     | 1                    | 675M    |
|                                    | 3.99                     | 2                    | 675M    |
|                                    | 2.51                     | 4                    | 675M    |
|                                    | 1.99                     | 8                    | 675M    |
| iSM (ours)                         | 5.27                     | 1                    | 675M    |
|                                    | 2.44                     | 2                    | 675M    |
|                                    | 2.05                     | 4                    | 675M    |
|                                    | 1.93                     | 8                    | 675M    |
|                                    | 1.88                     | 128                  | 675M    |

Table 2: **Broader evaluation** with FD-DINOv2 and IS on ImageNet  $256 \times 256$ .

| Model             | NFE      | FD-DINOv2 $\downarrow$ | IS $\uparrow$ |
|-------------------|----------|------------------------|---------------|
| SM [18]           | 1        | 500.92                 | 102.66        |
| IMM [69]          | 1        | 247.78                 | 128.87        |
| <b>iSM (ours)</b> | <b>1</b> | <b>232.31</b>          | <b>223.52</b> |
| SM [18]           | 2        | 329.53                 | 125.66        |
| IMM [69]          | 2        | 152.08                 | 173.66        |
| <b>iSM (ours)</b> | <b>2</b> | <b>107.63</b>          | <b>302.29</b> |
| SM [18]           | 4        | 265.90                 | 136.79        |
| IMM [69]          | 4        | 110.88                 | 204.95        |
| <b>iSM (ours)</b> | <b>4</b> | <b>83.70</b>           | <b>298.23</b> |

Table 3: **Ablation study** on ImageNet  $256 \times 256$ . We investigate the impact of key hyperparameters for each of our proposed components. The best-performing setting from each block is carried forward to the next. All models are trained for 250K iterations.

| Method                       | $w_{\max}$ | $t_{\text{interval}}$ | $L$      | $K$       | $\theta_{\text{target}}^-$ | FID $_{N=1} \downarrow$ | FID $_{N=4} \downarrow$ |
|------------------------------|------------|-----------------------|----------|-----------|----------------------------|-------------------------|-------------------------|
| Intrinsic Guidance (Sec. 3)  | 2.0        | 0                     | 0        | 0         | 0.9999                     | 10.10                   | 3.21                    |
|                              | <b>3.5</b> | 0                     | 0        | 0         | 0.9999                     | 9.62                    | 3.17                    |
|                              | 5.0        | 0                     | 0        | 0         | 0.9999                     | 10.38                   | 3.34                    |
| Interval Guidance (Sec. 3)   | 3.5        | 0.0                   | 0        | 0         | 0.9999                     | 9.62                    | 3.17                    |
|                              | 3.5        | 0.1                   | 0        | 0         | 0.9999                     | 8.58                    | 3.14                    |
|                              | 3.5        | <b>0.3</b>            | 0        | 0         | 0.9999                     | 8.49                    | 2.81                    |
|                              | 3.5        | 0.5                   | 0        | 0         | 0.9999                     | 19.22                   | 2.84                    |
| Multi-level Wavelet (Sec. 4) | 3.5        | 0.3                   | 0        | 0         | 0.9999                     | 8.49                    | 2.81                    |
|                              | 3.5        | 0.3                   | 1        | 0         | 0.9999                     | 8.21                    | 2.79                    |
|                              | 3.5        | 0.3                   | 3        | 0         | 0.9999                     | 8.17                    | 2.75                    |
|                              | 3.5        | 0.3                   | <b>5</b> | 0         | 0.9999                     | 8.12                    | 2.64                    |
| Scaling OT (Sec. 5)          | 3.5        | 0.3                   | 5        | 0         | 0.9999                     | 8.12                    | 2.64                    |
|                              | 3.5        | 0.3                   | 5        | 1         | 0.9999                     | 8.07                    | 2.51                    |
|                              | 3.5        | 0.3                   | 5        | 8         | 0.9999                     | 8.03                    | 2.28                    |
|                              | 3.5        | 0.3                   | 5        | <b>32</b> | 0.9999                     | 7.97                    | 2.23                    |
| Twin EMA (Sec. 6)            | 3.5        | 0.3                   | 5        | 32        | 0.9999                     | 7.97                    | 2.23                    |
|                              | 3.5        | 0.3                   | 5        | 32        | 0.999                      | 7.43                    | 2.21                    |
|                              | 3.5        | 0.3                   | 5        | 32        | <b>0.95</b>                | 6.56                    | 2.16                    |

FID of 5.27, *outperforming both*. This strong performance extends to few-step scenarios, with FIDs of 2.05 (4 steps) and 1.93 (8 steps), highlighting iSM’s robustness across different inference budgets.

To provide a more comprehensive validation, we extend our evaluation to include FD-DINOv2 and Inception Score (IS). As presented in Table 2, our iSM framework consistently outperforms the baselines across these metrics. The strong FD-DINOv2 scores confirm that our model’s high fidelity is robust and not specific to the Inception feature space; for example, at 4 NFE, iSM achieves a score of 83.70, a *greater than 3 times improvement* over the baseline SM’s 265.90. The Inception Scores show a particularly large margin of improvement; for example, at 2 inference steps, our model’s IS of 302.29 more than *doubles* the baseline score of 125.66. The consistent gains across these distinct evaluation metrics provide strong evidence that iSM yields significant and robust improvements to shortcut model generation.

## 7.2 Component-wise Analysis

We explore whether our proposed techniques enhance shortcut model training. As shown in Table 3, each component contributes to improved generation quality, with their combination achieving a significantly better FID score compared to the vanilla model. Below, we present a detailed analysis of the individual impact of each component.

**Intrinsic Guidance.** We examine the effect of varying the range of the conditioning CFG scale  $w$  on model performance. During training, we sample discrete values of  $w$  from the interval  $[0, w_{\max}]$  using a step size of 0.25. Through empirical evaluation, we find that setting  $w_{\max} = 3.5$  leads to the best results. When  $w_{\max}$  is set too high (e.g., 5.0), the model is exposed to a broader range of conditioning strengths. This can introduce unnecessary complexity, as the model must learn to generalize over a wider set of conditioning values, potentially degrading overall performance. On the other hand, if  $w_{\max}$  is too low (e.g., 2.0), the model only learns from outputs corresponding to low CFG scales, which may lack high quality.

**Interval Guidance in Training.** We examined the effect of varying  $t_{\text{interval}}$ , the threshold below which CFG is disabled. We observed that increasing  $t_{\text{interval}}$  from very low values up to a moderate range (e.g., 0.1 to 0.3) improved both metrics, consistent with the benefit of avoiding strong, coarse guidance in high-noise regimes. However, further increasing  $t_{\text{interval}}$  to higher values (e.g., 0.5) results in a significant increase in FID. Training the model without guidance for a substantial portion of the early reverse process may impair its ability to effectively leverage guidance when  $t \geq t_{\text{interval}}$ .

**Multi-level Wavelet Function.** We also investigate the impact of varying the number of decomposition levels in the wavelet transform. Given that the latent representation is of size  $32 \times 32$ , the maximum feasible number of levels is  $\log_2(32) = 5$ . Our experiments show that utilizing the full 5

levels yields better performance compared to using fewer levels (e.g., 1 or 3) or not using the wavelet transform entirely.

**Scaling Optimal Transport.** We also examine the impact of OT matching on the performance of our previous improvements. Our results show that using traditional OT matching results in lower FID scores than not using OT. We then assess the effectiveness of our proposed enhanced OT by applying it at two different scales:  $K = 8$  and  $K = 32$ . Our results reveal a clear correlation between increasing OT scale and improved FID scores. Although scaling beyond  $K = 32$  may further enhance performance, we select  $K = 32$  as a trade-off between computational overhead and effectiveness.

**Twin EMA.** Finally, our proposed Twin EMA was evaluated using decay rates of 0.999 and 0.95 for the target network  $\theta_{\text{target}}^-$ . A decay rate of 0.95 provided an effective balance between training stability and mitigating distributional divergence from self-consistency targets. This was evidenced by a substantial reduction in FID, achieving 6.56 and 2.16 for one-step and four-step generation, respectively.

### 7.3 Generalizing to Higher Resolutions and Architectures

To demonstrate the *architectural generality* and *resolution scalability* of our framework, we applied our **iSM** training to FlowDCN [59], a fully convolutional generative model with group-wise deformable convolution blocks, on ImageNet  $512 \times 512$ . As shown in Table 4, at 300K training iterations, our method yields a substantial FID improvement over the baseline, confirming its effectiveness across different model families and at higher scales. For instance, in 4-step generation, our approach reduces the FID from 12.16 to 9.94. The improvement is also pronounced in the one-step generation, where iSM improves sample diversity by *boosting Recall over 5 times* from 0.11 to 0.55.

Table 4: **Scalability and generality** of iSM on ImageNet  $512 \times 512$  with the FlowDCN architecture, reported after 300K training iterations.

| Model             | NFE | FID↓         | Precision↑  | Recall↑     |
|-------------------|-----|--------------|-------------|-------------|
| SM                | 1   | 43.81        | 0.56        | 0.11        |
| <b>iSM (ours)</b> | 1   | <b>37.05</b> | <b>0.60</b> | <b>0.55</b> |
| SM                | 4   | 12.16        | <b>0.86</b> | 0.19        |
| <b>iSM (ours)</b> | 4   | <b>9.94</b>  | 0.78        | <b>0.62</b> |

## 8 Conclusion

This work confronts the foundational obstacles that have previously limited the performance of shortcut models. We systematically address five core challenges: the damaging effects of compounding guidance, inflexible fixed guidance, a pervasive low-frequency bias, conflicts between self-consistency and EMA updates, and instability from high-curvature flow trajectories. To resolve these issues, we present **iSM**, a unified training framework composed of four key components. By incorporating *Intrinsic Guidance* for dynamic control while resolving the flaw of compounding guidance, a *Multi-Level Wavelet Loss* for high-frequency fidelity, *Scaling Optimal Transport (sOT)* for smoother trajectories, and a *Twin EMA* strategy for stable consistency, iSM elevates shortcut models into a competitive paradigm for generative modeling.

Our extensive experiments on ImageNet  $256 \times 256$  demonstrate that iSM yields substantial FID improvements over baseline shortcut models across one-step, few-step, and multi-step generation. These results close the performance gap with leading GANs and other diffusion-based models, establishing shortcut models as a viable and competitive alternative that uniquely offers variable sampling budgets from a single network.

## References

- [1] S. AI. Fine-tuned vae for stable diffusion (sd-vae-ft-mse). <https://huggingface.co/stabilityai/sd-vae-ft-mse>, 2023. Accessed: 2025-04-28.
- [2] F. Bao, S. Nie, K. Xue, Y. Cao, C. Li, H. Su, and J. Zhu. All are worth words: A vit backbone for diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22669–22679, 2023.
- [3] R. Basri, M. Galun, A. Geifman, D. Jacobs, Y. Kasten, and S. Kritchman. Frequency bias in neural networks for input of non-uniform density. In *International conference on machine learning*, pages 685–694. PMLR, 2020.
- [4] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.
- [5] A. Brock. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.
- [6] H. Chang, H. Zhang, L. Jiang, C. Liu, and W. T. Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325, 2022.
- [7] C. Chen, R. Qian, W. Hu, T.-J. Fu, J. Tong, X. Wang, L. Li, B. Zhang, A. Schwing, W. Liu, et al. Dit-air: Revisiting the efficiency of diffusion model architecture design in text to image generation. *arXiv preprint arXiv:2503.10618*, 2025.
- [8] J. Chen, S. Xue, Y. Zhao, J. Yu, S. Paul, J. Chen, H. Cai, E. Xie, and S. Han. Sana-sprint: One-step diffusion with continuous-time consistency distillation, 2025.
- [9] J. Chen, J. Yu, C. Ge, L. Yao, E. Xie, Y. Wu, Z. Wang, J. Kwok, P. Luo, H. Lu, et al. Pixart- $\alpha$ : Fast training of diffusion transformer for photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426*, 2023.
- [10] Y. Chen, G. Li, C. Jin, S. Liu, and T. Li. Ssd-gan: Measuring the realness in the spatial and spectral domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1105–1112, 2021.
- [11] T. Dao, T. H. Nguyen, T. Le, D. Vu, K. Nguyen, C. Pham, and A. Tran. Swiftbrush v2: Make your one-step diffusion model better than its teacher. In *European Conference on Computer Vision*, pages 176–192. Springer, 2024.
- [12] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.
- [13] P. Dvurechensky, A. Gasnikov, and A. Kroshnin. Computational optimal transport: Complexity by accelerated gradient descent is better than by sinkhorn’s algorithm. In *International conference on machine learning*, pages 1367–1376. PMLR, 2018.
- [14] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.
- [15] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.
- [16] P. Esser, R. Rombach, and B. Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.
- [17] J. Frank, T. Eisenhofer, L. Schönherr, A. Fischer, D. Kolossa, and T. Holz. Leveraging frequency analysis for deep fake image recognition. In *International conference on machine learning*, pages 3247–3258. PMLR, 2020.

- [18] K. Frans, D. Hafner, S. Levine, and P. Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.
- [19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.
- [20] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6626–6637, 2017.
- [21] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.
- [22] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [23] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022.
- [24] J. Ho and T. Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.
- [25] E. Hoogeboom, J. Heek, and T. Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*, pages 13213–13232. PMLR, 2023.
- [26] M. Kang, J.-Y. Zhu, R. Zhang, J. Park, E. Shechtman, S. Paris, and T. Park. Scaling up gans for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10124–10134, 2023.
- [27] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.
- [28] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.
- [29] M. Khayatkhoei and A. Elgammal. Spatial frequency bias in convolutional generative adversarial networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7152–7159, 2022.
- [30] D. Kingma, T. Salimans, B. Poole, and J. Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.
- [31] T. Kynkäänniemi, M. Aittala, T. Karras, S. Laine, T. Aila, and J. Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models. *arXiv preprint arXiv:2404.07724*, 2024.
- [32] B. F. Labs. Flux. <https://github.com/black-forest-labs/flux>, 2024.
- [33] S. Lee, B. Kim, and J. C. Ye. Minimizing trajectory curvature of ode-based generative models. In *International Conference on Machine Learning*, pages 18957–18973. PMLR, 2023.
- [34] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos. Mmd gan: Towards deeper understanding of moment matching network. *Advances in neural information processing systems*, 30, 2017.
- [35] T. Li, Y. Tian, H. Li, M. Deng, and K. He. Autoregressive image generation without vector quantization. *arXiv preprint arXiv:2406.11838*, 2024.



- [36] Y. Li, K. Swersky, and R. Zemel. Generative moment matching networks. In *International conference on machine learning*, pages 1718–1727. PMLR, 2015.
- [37] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.
- [38] X. Liu, C. Gong, and Q. Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.
- [39] C. Lu and Y. Song. Simplifying, stabilizing and scaling continuous-time consistency models. *arXiv preprint arXiv:2410.11081*, 2024.
- [40] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.
- [41] N. Ma, M. Goldstein, M. S. Albergo, N. M. Boffi, E. Vanden-Eijnden, and S. Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. *arXiv preprint arXiv:2401.08740*, 2024.
- [42] T. H. Nguyen and A. Tran. Swiftbrush: One-step text-to-image diffusion model with variational score distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7807–7816, 2024.
- [43] OpenAI. Video generation models as world simulators. <https://openai.com/sora/>, 2024.
- [44] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.
- [45] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.
- [46] A.-A. Pooladian, H. Ben-Hamu, C. Domingo-Enrich, B. Amos, Y. Lipman, and R. T. Chen. Multisample flow matching: Straightening flows with minibatch couplings. In *International Conference on Machine Learning*, pages 28100–28127. PMLR, 2023.
- [47] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pages 5301–5310. PMLR, 2019.
- [48] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.
- [49] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.
- [50] K. Schwarz, Y. Liao, and A. Geiger. On the frequency bias of generative models. *Advances in Neural Information Processing Systems*, 34:18126–18136, 2021.
- [51] K. Schwarz, Y. Liao, and A. Geiger. On the frequency bias of generative models. *Advances in Neural Information Processing Systems*, 34:18126–18136, 2021.
- [52] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.
- [53] Y. Song and P. Dhariwal. Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*, 2023.
- [54] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

- [55] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.
- [56] K. Tian, Y. Jiang, Z. Yuan, B. Peng, and L. Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *arXiv preprint arXiv:2404.02905*, 2024.
- [57] Y. Tian, Z. Tu, H. Chen, J. Hu, C. Xu, and Y. Wang. U-dits: Downsample tokens in u-shaped diffusion transformers. *arXiv preprint arXiv:2405.02730*, 2024.
- [58] T. Wan, A. Wang, B. Ai, B. Wen, C. Mao, C.-W. Xie, D. Chen, F. Yu, H. Zhao, J. Yang, J. Zeng, J. Wang, J. Zhang, J. Zhou, J. Wang, J. Chen, K. Zhu, K. Zhao, K. Yan, L. Huang, M. Feng, N. Zhang, P. Li, P. Wu, R. Chu, R. Feng, S. Zhang, S. Sun, T. Fang, T. Wang, T. Gui, T. Weng, T. Shen, W. Lin, W. Wang, W. Wang, W. Zhou, W. Wang, W. Shen, W. Yu, X. Shi, X. Huang, X. Xu, Y. Kou, Y. Lv, Y. Li, Y. Liu, Y. Wang, Y. Zhang, Y. Huang, Y. Li, Y. Wu, Y. Liu, Y. Pan, Y. Zheng, Y. Hong, Y. Shi, Y. Feng, Z. Jiang, Z. Han, Z.-F. Wu, and Z. Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv: 2503.20314*, 2025.
- [59] S. Wang, Z. Li, T. Song, X. Li, T. Ge, B. Zheng, and L. Wang. Exploring DCN-like architecture for fast image generation with arbitrary resolution. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [60] S. Wang, Z. Li, T. Song, X. Li, T. Ge, B. Zheng, and L. Wang. Flowdcn: Exploring dcn-like architectures for fast image generation with arbitrary resolution. *arXiv preprint arXiv:2410.22655*, 2024.
- [61] E. Xie, J. Chen, J. Chen, H. Cai, H. Tang, Y. Lin, Z. Zhang, M. Li, L. Zhu, Y. Lu, and S. Han. Sana: Efficient high-resolution image synthesis with linear diffusion transformer, 2024.
- [62] Z.-Q. J. Xu, Y. Zhang, T. Luo, Y. Xiao, and Z. Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019.
- [63] Z.-Q. J. Xu, Y. Zhang, and Y. Xiao. Training behavior of deep neural network in frequency domain. In *Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part I* 26, pages 264–274. Springer, 2019.
- [64] J. Yao, C. Wang, W. Liu, and X. Wang. Fasterdit: Towards faster diffusion transformers training without architecture modification. *Advances in Neural Information Processing Systems*, 37:56166–56189, 2024.
- [65] T. Yin, M. Gharbi, T. Park, R. Zhang, E. Shechtman, F. Durand, and B. Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in neural information processing systems*, 37:47455–47487, 2024.
- [66] T. Yin, M. Gharbi, R. Zhang, E. Shechtman, F. Durand, W. T. Freeman, and T. Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6613–6623, 2024.
- [67] S. Yu, S. Kwak, H. Jang, J. Jeong, J. Huang, J. Shin, and S. Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations*, 2025.
- [68] Y. Zhang and B. Hooi. Hipa: enabling one-step text-to-image diffusion models via high-frequency-promoting adaptation. *arXiv preprint arXiv:2311.18158*, 2023.
- [69] L. Zhou, S. Ermon, and J. Song. Inductive moment matching. *arXiv preprint arXiv:2503.07565*, 2025.

## NeurIPS Paper Checklist

### 1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: Our contributions are accurately reflected in the abstract and in the summary of introduction section Section 1.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

### 2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: We explicitly mention the limitation for future works in Section 8.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

### 3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: N/A, because our work focuses solely on empirical evaluation to measure the effectiveness of the proposed framework.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

#### 4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: Yes, this information can be found in Section 7 and in Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

#### 5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Yes, we will release our code upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

## 6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Yes, we provided them in Section 7 and in Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

## 7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: No, because the evaluation process takes hours on a GPU and is very expensive to compute with multiple seeds. Importantly, we conducted evaluation for several benchmarks in Section 7 so running with different seeds are not feasible in our scope.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

#### 8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: Yes, we provided them in the Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

#### 9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [\[Yes\]](#)

Justification: Yes, we totally confirm to it.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

#### 10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[Yes\]](#)

Justification: We discussed the societal impacts in Section 8.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

## 11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [\[Yes\]](#)

Justification: Yes, our code will be released with Licences.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

## 12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [\[Yes\]](#)

Justification: Yes, we have properly cited the code, data and models used.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, [paperswithcode.com/datasets](https://paperswithcode.com/datasets) has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

### 13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: At the time of submission, no new assets are being released. We plan to release the associated code publicly if the paper is accepted.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

### 14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

### 15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.



#### 16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The paper does not involve LLM in the core of this method.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

## A Proof of Proposition 1

Given a shortcut model  $s_\theta$  and its corresponding classifier-free guided output  $g_\theta^w$  with guidance scale  $w$ . To analyze the generation process, we define a sequence of intermediate points  $\{\mathbf{x}'_{\frac{i}{N}}\}_{i=0}^N$ . This sequence is constructed by recursively applying  $N$  consecutive shortcut steps, each with the smallest step size  $d = 1/N$ . Note that  $N$  is chosen to be a power of 2. The process begins from the initial noise  $\mathbf{x}_0$  at  $t = 0$ , conditioned on  $c$ . The starting point of the sequence is thus defined as  $\mathbf{x}'_0 = \mathbf{x}_0$ , and subsequent intermediate points are generated as follow:

$$\mathbf{x}'_{\frac{i+1}{N}} = \mathbf{x}'_{\frac{i}{N}} + s_\theta\left(\mathbf{x}'_{\frac{i}{N}}, \frac{i}{N}, c, d\right) d, \quad \text{for } i = 0, \dots, N-1. \quad (13)$$

Ideally, we assume that the model is perfectly trained, or equivalently, that the loss in Eq. (4) is minimized to zero. Under this assumption, we have:

$$s_\theta(\mathbf{x}'_t, t, c, 2d) = \frac{1}{2} [g_\theta^w(\mathbf{x}'_t, t, c, d) + g_\theta^w(\mathbf{x}'_{t+d}, t+d, c, d)], \quad (14)$$

$$s_\theta(\mathbf{x}'_t, t, \emptyset, 2d) = \frac{1}{2} [s_\theta(\mathbf{x}'_t, t, \emptyset, d) + s_\theta(\mathbf{x}'_{t+d}, t+d, \emptyset, d)]. \quad (15)$$

In the following derivation, we omit the time  $t$  in the network notation for simplicity, unless otherwise specified. First, we will prove by induction that:

$$g_\theta^w(\mathbf{x}'_t, c, 2^j d) = \frac{1}{2^j} \sum_{i=0}^{2^j-1} g_\theta^{w^{j+1}}(\mathbf{x}'_{t+id}, c, d). \quad (16)$$

For the base case  $j = 0$ , we have  $g_\theta^w(\mathbf{x}'_t, c, d) = g_\theta^w(\mathbf{x}'_t, c, d)$ , which satisfies Eq. (16). Assuming that Eq. (16) holds for  $j = k$ , we now show that it also holds for  $j = k+1$ . Using Eq. (14) and Eq. (15), we obtain:

$$\begin{aligned} s_\theta(\mathbf{x}'_t, c, 2^{k+1}d) &= \frac{1}{2} [g_\theta^w(\mathbf{x}'_t, c, 2^k d) + g_\theta^w(\mathbf{x}'_{t+2^k d}, c, 2^k d)] \\ &= \frac{1}{2^{k+1}} \left[ \sum_{i=0}^{2^k-1} g_\theta^{w^{k+1}}(\mathbf{x}'_{t+id}, c, d) + \sum_{i=0}^{2^k-1} g_\theta^{w^{k+1}}(\mathbf{x}'_{t+2^k d+id}, c, d) \right] \\ &= \frac{1}{2^{k+1}} \sum_{i=0}^{2^{k+1}-1} g_\theta^{w^{k+1}}(\mathbf{x}'_{t+id}, c, d), \\ s_\theta(\mathbf{x}'_t, \emptyset, 2^{k+1}d) &= \frac{1}{2} [s_\theta(\mathbf{x}'_t, \emptyset, 2^k d) + s_\theta(\mathbf{x}'_{t+2^k d}, \emptyset, 2^k d)] \\ &= \frac{1}{2^{k+1}} \sum_{i=0}^{2^{k+1}-1} s_\theta(\mathbf{x}'_{t+id}, \emptyset, d). \end{aligned}$$

Therefore, we have:

$$\begin{aligned} g_\theta^w(\mathbf{x}'_t, c, 2^{k+1}d) &= w s_\theta(\mathbf{x}'_t, c, 2^{k+1}d) + (1-w) s_\theta(\mathbf{x}'_t, \emptyset, 2^{k+1}d) \\ &= \frac{1}{2^{k+1}} \sum_{i=0}^{2^{k+1}-1} [w g_\theta^{w^{k+1}}(\mathbf{x}'_{t+id}, c, d) + (1-w) s_\theta(\mathbf{x}'_{t+id}, \emptyset, d)] \\ &= \frac{1}{2^{k+1}} \sum_{i=0}^{2^{k+1}-1} [w(w^{k+1} s_\theta(\mathbf{x}'_{t+id}, c, d) + (1-w^{k+1}) s_\theta(\mathbf{x}'_{t+id}, \emptyset, d)) + (1-w) s_\theta(\mathbf{x}'_{t+id}, \emptyset, d)] \\ &= \frac{1}{2^{k+1}} \sum_{i=0}^{2^{k+1}-1} [w^{k+2} s_\theta(\mathbf{x}'_{t+id}, c, d) + (1-w^{k+2}) s_\theta(\mathbf{x}'_{t+id}, \emptyset, d)] \\ &= \frac{1}{2^{k+1}} \sum_{i=0}^{2^{k+1}-1} g_\theta^{w^{k+2}}(\mathbf{x}'_{t+id}, c, d). \end{aligned}$$

This satisfies Eq. (16) for  $j = k + 1$ . By induction, we have Eq. (16) holds for all  $j$ . Based on this result, we can rewrite the output of the shortcut model for a single large step of size  $Nd = 1$ , denoted by  $s_\theta(x_0, c, Nd)$ , as follows:

$$\begin{aligned} s_\theta(x_0, c, Nd) &= \frac{1}{2} \left[ g_\theta^w \left( x_0, c, \frac{N}{2}d \right) + g_\theta^w \left( x'_{\frac{1}{2}}, c, \frac{N}{2}d \right) \right] \\ &= \frac{1}{N} \left[ \sum_{i=0}^{N/2-1} g_\theta^{w^{\log_2(N/2)+1}} \left( x'_{\frac{i}{N}}, c, d \right) + \sum_{i=0}^{N/2-1} g_\theta^{w^{\log_2(N/2)+1}} \left( x'_{\frac{1}{2} + \frac{i}{N}}, c, d \right) \right] \\ &= \frac{1}{N} \sum_{i=0}^{N-1} g_\theta^{w^{\log_2(N)}} \left( x'_{\frac{i}{N}}, c, d \right), \end{aligned}$$

which completes the proof.

## B Multi-Level Wavelet Function

**Multi-Level Wavelet Function.** Algorithm 1 provides pseudo-code for our proposed multi-level wavelet objective described in Section 4.

---

### Algorithm 1 Multi-level Wavelet Function

---

```
class MultiLevelWaveletLoss:
    def __init__(self):
        self.dwt = DWT_2D("haar") # Discrete Wavelet Transform (DWT)
        self.diff_func = MSELoss() # Distance function

    def concatenated_dwt(self, x):
        xll, xlh, xhl, xhh = self.dwt(x) # Decompose into 4 wavelet sub-bands using DWT
        details = torch.cat([xll, xlh, xhl, xhh], dim=1) # Concatenate the sub-bands
        return details

    def __call__(self, pred, target, num_levels):
        total_loss = diff_func(pred, target).mean() # Calculate low-level loss in original output space

        # Recursively calculate loss on wavelet sub-bands for each level
        pred_curr, target_curr = pred, target
        for current_level in range(num_levels):
            # Derive predicted and target sub-bands using outputs from previous level
            pred_bands = self.concatenated_dwt(pred_curr)
            target_bands = self.concatenated_dwt(target_curr)

            # Calculate loss on current level
            total_loss += diff_func(pred_bands, target_bands).mean()
            pred_curr, target_curr = pred_bands, target_bands

        # Taking average from all levels
        loss = total_loss / (num_levels + 1)
        return loss
```

---

**Results.** Fig. 3 presents qualitative comparisons illustrating the impact of different levels in our multi-level wavelet function versus traditional low-level loss. Incorporating more wavelet levels yields finer details and fewer artifacts, especially in one- and few-step generation.

## C Injecting Conditional Inputs into Network

We explore two primary strategies for incorporating conditional information including CFG scale  $w$ , current sample  $x_t$ , time  $t$ , condition  $c$ , and the target step size  $d$  into our network. The first strategy, similar to U-ViT [2], encodes each condition as an individual token appended to the input sequence of noisy image patch tokens. The second strategy employs AdaLN-Zero blocks [44] to modulate the network with each condition; these modulations are then aggregated through addition. We find that the AdaLN-Zero approach yields comparable performance to the U-ViT method but without the drawback of increased input sequence length. Given this efficiency benefit, we adopt AdaLN-Zero for injecting CFG information into our network.

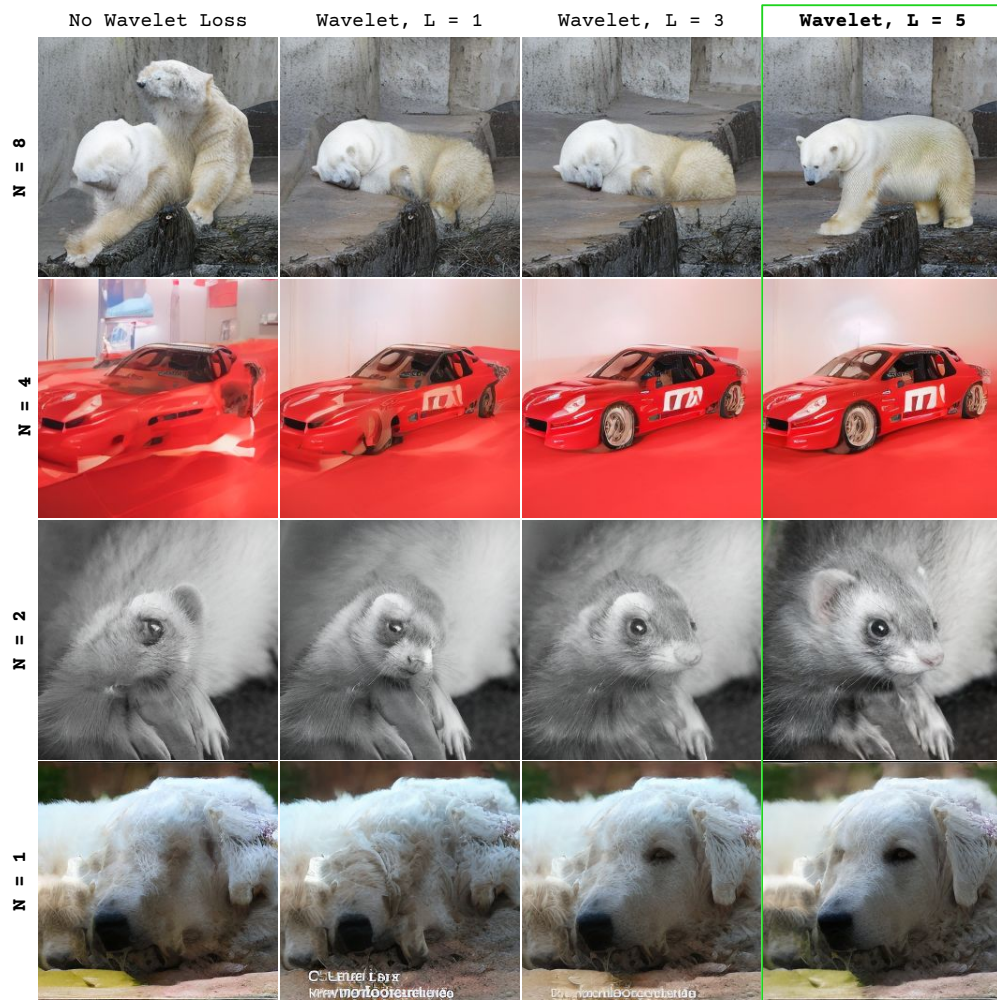


Figure 3: Effect of applying different numbers of wavelet loss layers ( $L$ ) on image quality under one and few-step inference. The traditional low-level objective (leftmost column) results in noticeable degradation and artifacts. In contrast, using multi-level wavelet loss, especially with  $L = 5$  (green box, right), produces high-quality, consistent images across all inference settings.

## D Accelerating training speed

To mitigate the high computational cost of shortcut models, we explore resuming their training from existing a flow matching weights [67]. Table 5 shows that while resuming offers limited improvement for the original shortcut model, our improved shortcut models (iSM) demonstrate considerable gains. Specifically, when incorporating a series of proposed methods, iSM achieves significantly better FID scores. Models are trained for 250k steps for all evaluations.

Table 5: Performance of Improved Shortcut Models

| Method                                | FID <sub>N=1</sub> ↓ | FID <sub>N=4</sub> ↓ |
|---------------------------------------|----------------------|----------------------|
| Shortcut Models [18]                  | 21.38                | 13.46                |
| <i>Improved Shortcut Models (iSM)</i> |                      |                      |
| + Intrinsic Guidance                  | 9.62                 | 3.17                 |
| + Interval Guidance in Training       | 8.49                 | 2.81                 |
| + Multi-level Wavelet Function        | 8.12                 | 2.64                 |
| + Scaling Optimal Transport           | 7.97                 | 2.23                 |
| + Twin EMA                            | 6.56                 | 2.16                 |

## E Experiment Settings

### E.1 Training & Parameterization Setting

For experiments on ImageNet, images are preprocessed to 256x256 resolution, following the protocol of ADM [12]. Our models operate in the latent space derived from the sd-vae-ft-mse autoencoder [1]. These latents are subsequently normalized channel-wise using means of [0.86488, -0.27787343, 0.21616915, 0.3738409] and standard deviations of [4.85503674, 5.31922414, 3.93725398, 3.9870003]. Finally, the normalized latents are scaled

Table 6: Experimental settings on ImageNet  $256 \times 256$ .

| Parameterization Setting                          |                         |
|---------------------------------------------------|-------------------------|
| Architecture                                      | SiT-XL                  |
| GFlops                                            | 118.64                  |
| Params (M)                                        | 675                     |
| Flow Trajectory                                   | OT-FM                   |
| Input dim.                                        | $32 \times 32 \times 4$ |
| Num. layers                                       | 24                      |
| Hidden dim.                                       | 1024                    |
| Num. heads                                        | 16                      |
| $\alpha_t$                                        | $1 - t$                 |
| $\sigma_t$                                        | $t$                     |
| $w_t$                                             | $\sigma_t$              |
| Training objective                                | v-prediction            |
| Training Setting                                  |                         |
| Training iteration                                | 800K                    |
| Dropout                                           | 0                       |
| Optimizer                                         | AdamW                   |
| AdamW $\beta_1$                                   | 0.9                     |
| AdamW $\beta_2$                                   | 0.999                   |
| AdamW $\epsilon$                                  | $10^{-8}$               |
| Learning Rate                                     | 0.0001                  |
| Weight Decay                                      | 0                       |
| Batch Size                                        | 256                     |
| Label Dropout                                     | 0.1                     |
| Methods                                           |                         |
| CFG Scale $w_{\max}$                              | 3.5                     |
| Interval $t_{\text{interval}}$                    | 0.3                     |
| Wavelet Levels $L$                                | 5                       |
| OT Scale $K$                                      | 32                      |
| Ratio of Empirical to Self-consistency Targets    | 0.25                    |
| EMA Parameters Used For Self-consistency Targets? | True                    |
| EMA Target Rate $\theta_{\text{target}}^-$        | 0.95                    |
| EMA Parameters Used For Evaluation?               | True                    |
| EMA Inference Rate $\theta_{\text{infer}}^-$      | 0.9999                  |

by a factor of 0.5, targeting an approximate standard deviation of 0.5 for the network input. Detailed configurations for all experiments are provided in Table 6.

## E.2 Evaluation Details

We follow the ADM [12] evaluation setup, using the same reference batches from their official implementation,<sup>4</sup> and compute FID [20] over 50K generated images.

## E.3 Baselines

Below, we summarize the baseline methods used in our evaluation.

- **ADM** [12] improves U-Net-based diffusion architectures and introduces classifier-guided sampling to balance sample quality and diversity.
- **CDM** [23] proposes cascaded diffusion models, which generate images in a coarse-to-fine manner, similar to ProgressiveGAN [27].

<sup>4</sup><https://github.com/openai/guided-diffusion/tree/main/evaluations>

- **Simple diffusion** [25] leverages a diffusion model for high-resolution images by simplifying the noise schedule and model architectures.
- **LDM** [48] introduces the concept of operating diffusion models in a compressed latent space, enhancing efficiency.
- **U-DiT** [57] proposes a series of U-shaped DiTs based on self-attention with downsampled tokens.
- **U-ViT** [2] adapts Vision Transformers for latent diffusion by incorporating U-Net-like long skip connections.
- **DiT** [44] pioneers the use of a pure transformer architecture as the backbone for diffusion models, featuring AdaIN-zero modules.
- **SiT** [41] investigates how to improve DiT training by transitioning from discrete diffusion to continuous flow-based modeling.
- **REPA** [67] accelerates the training of DiT/SiT models by regularizing network representations to align with features from pretrained visual encoders.
- **FlowDCN** [60] offers a fully convolutional architecture for generative modeling with linear time and memory complexity, enabling efficient high-resolution image synthesis.
- **iCT** [53] introduces several techniques to enhance the training of CMs [54], including a log-normal noise schedule, Pseudo-Huber loss functions, and a scheduler for total discretization steps during training.
- **SM** [18] establishes a framework for one-step generation by combining flow matching with a self-consistency objective.
- **IMM** [69] few-step generative models by inductively matching all moments of bootstrapped samples derived from stochastic interpolants using Maximum Mean Discrepancy (MMD), aiming for stable convergence.

## F Related Works

### F.1 Diffusion, Consistency & Flow Matching models

Generative modeling has seen significant advancements with methods like diffusion models, consistency models, and flow matching. Diffusion models [52, 55, 22, 30] are a generative framework that synthesizes data by gradually transforming random noise into outputs via a stochastic denoising process. To introduce efficient generation, consistency models [54, 53, 39] aim for efficient generation by learning a mapping from any point on a solution trajectory directly to the data origin. Meanwhile, flow matching [37, 38] proposes learning generative models by defining a target vector field between noise and data and training a neural network to approximate this field. Both these methods have demonstrated strong performance when scaled to text-to-image [48, 49, 45, 9, 15] and text-to-video [21, 4, 43] applications.

### F.2 End-to-end training for one-to-many step generative models

The quest for efficient, high-quality few-step generation has explored several avenues. While early one-step models leveraged GANs [19, 28, 5] and MMD [36, 34], the inherent instability and complexity of adversarial training limit their scalability. Consistency Models (CMs) [54, 53, 39] address the instability of adversarial training without needing synthetic datasets. Unlike distillation, CMs can be trained from scratch via consistency training (CT), independent of pre-trained diffusion models. However, few-step generation with CMs usually involves discrete-time variants. These variants require meticulous timestep scheduling and are susceptible to irreducible bias accumulation due to the inherent ambiguity of discretization. Inductive Moment Matching (IMM) [69] further advanced stable few-step training using moment matching. Shortcut Models (SMs) [18] introduced step-size conditioning with self-bootstrapping. However, original SMs (Section 3) suffer from inflexible guidance and guidance accumulation artifacts, which our work directly addresses.

## G More Qualitative Results



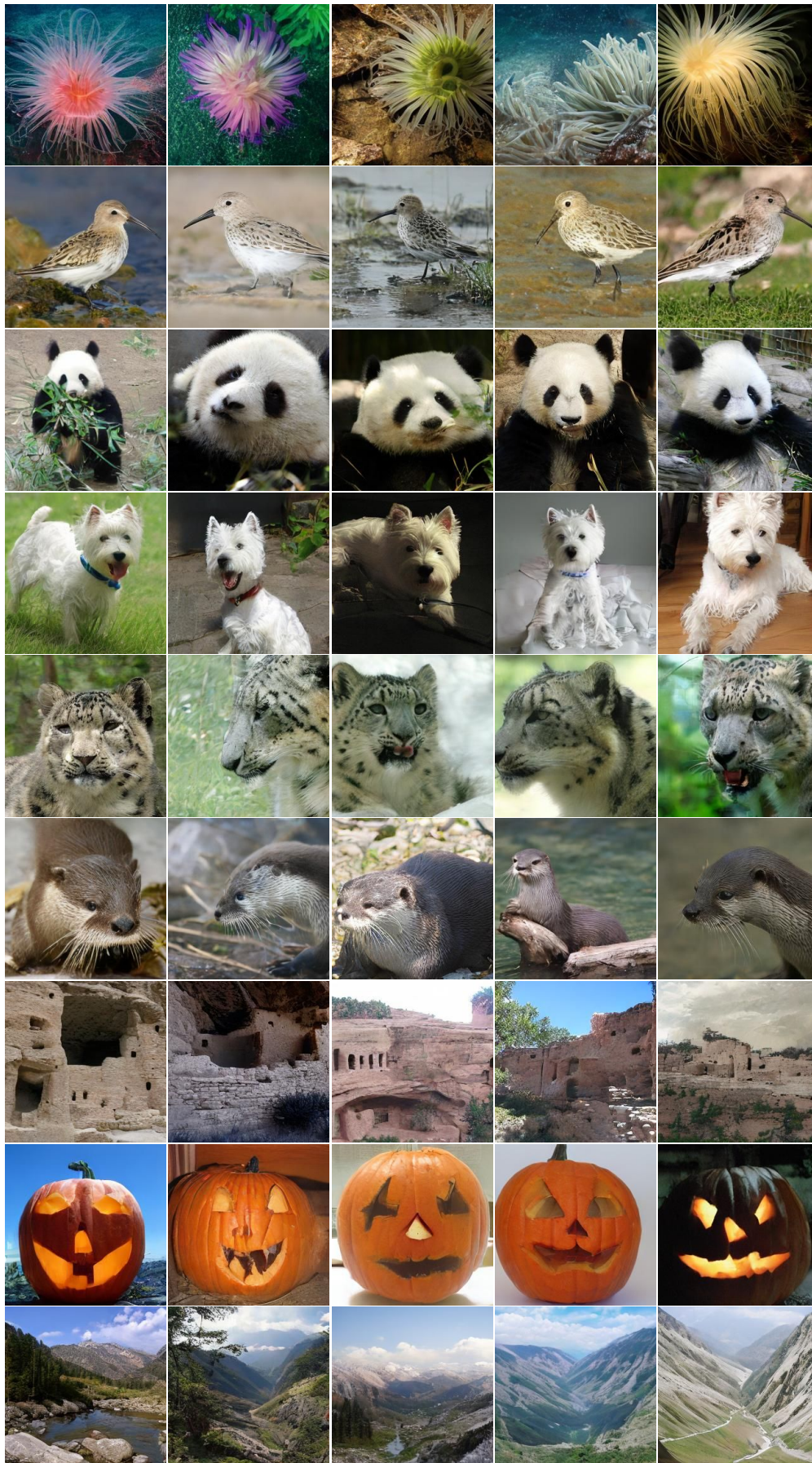


Figure 4: Uncurated samples on ImageNet-256  $\times$  256 with 1-step sampling.





Figure 5: Uncurated samples on ImageNet-256  $\times$  256 with 2-step sampling.





Figure 6: Uncurated samples on ImageNet-256  $\times$  256 with 4-step sampling.



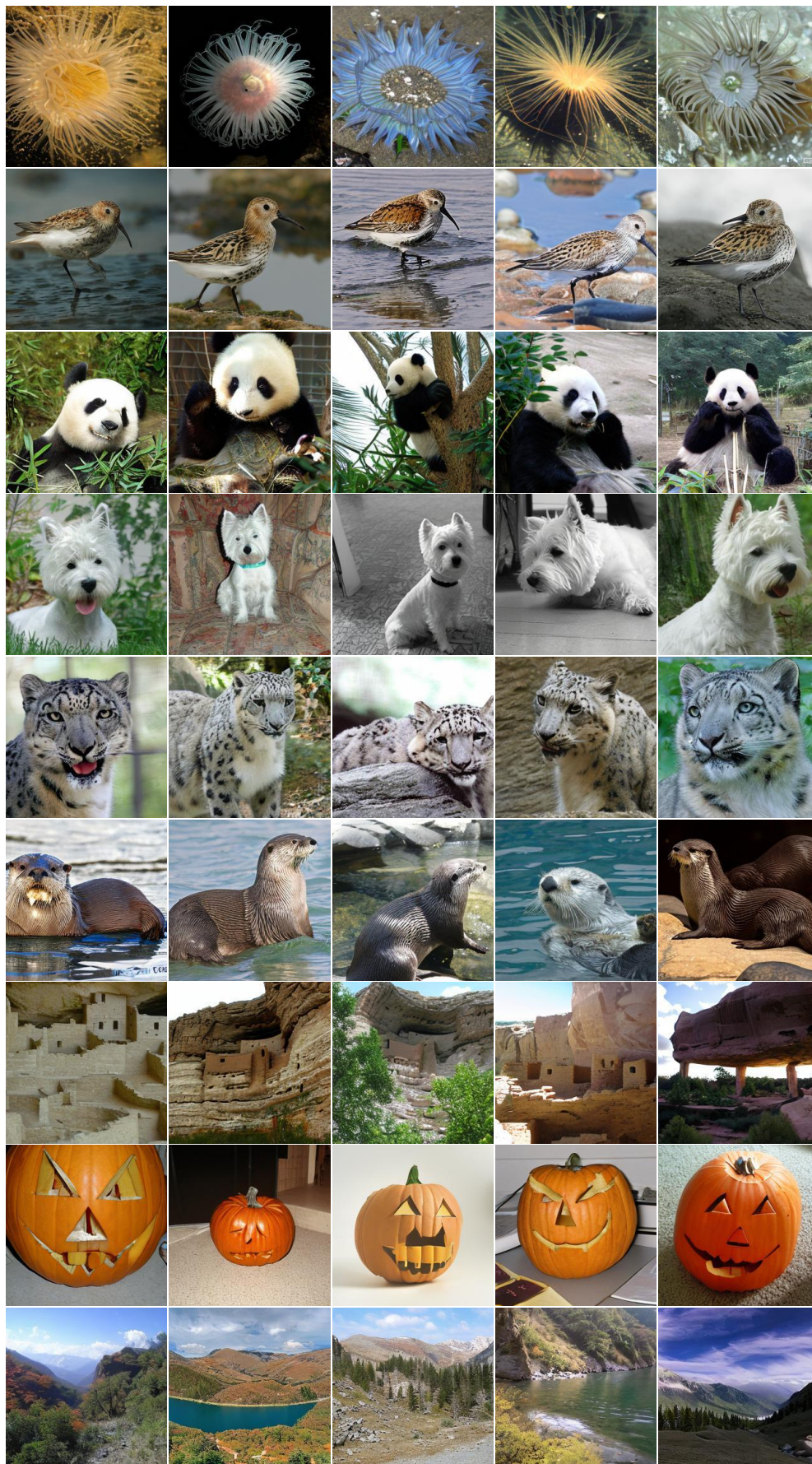


Figure 7: Uncurated samples on ImageNet-256  $\times$  256 with 8-step sampling.



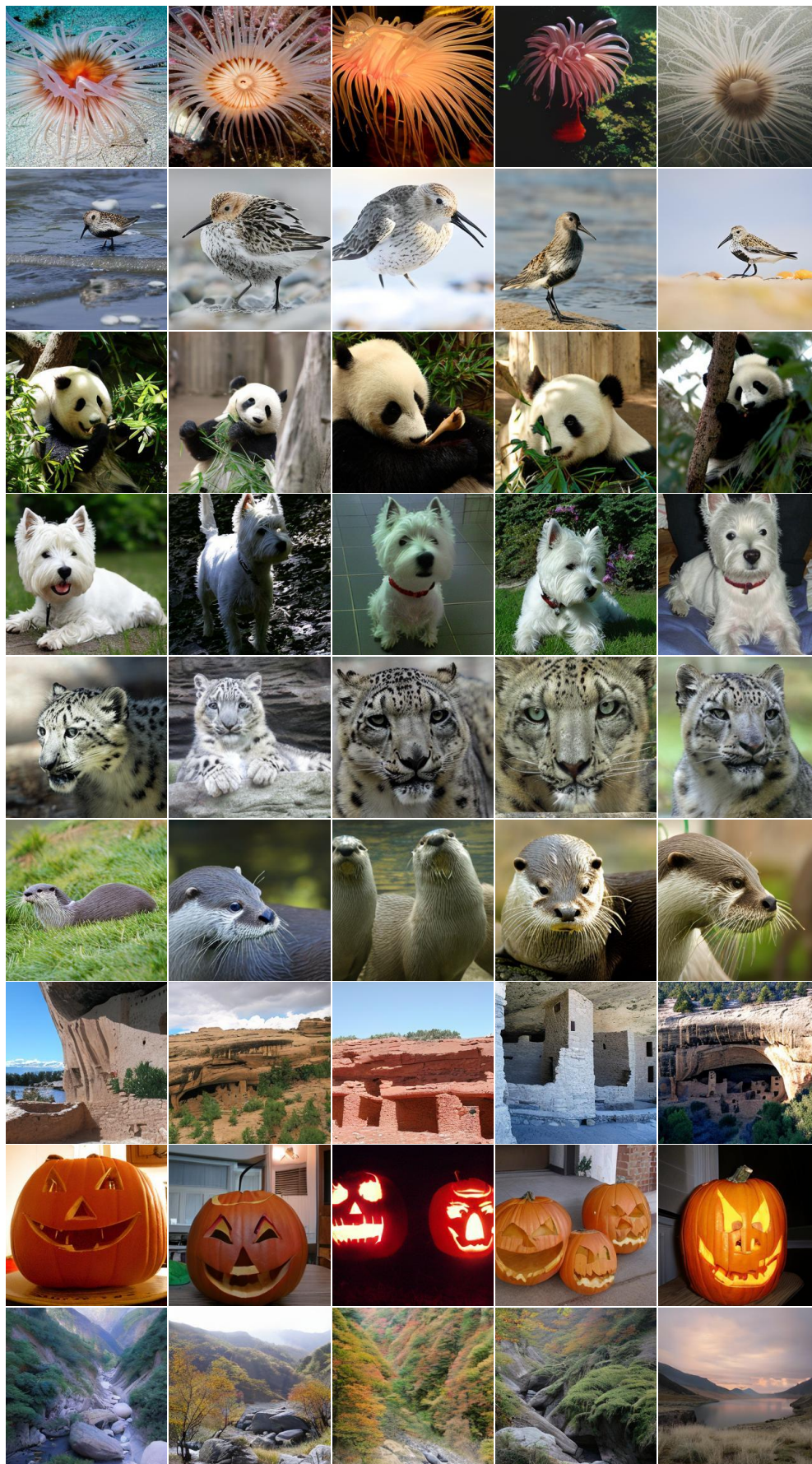


Figure 8: Uncurated samples on ImageNet-256  $\times$  256 with 128-step sampling.