Adaptive Discretization for Consistency Models

Jiayu Bai[1], **Zhanbo Feng**[2], **Zhijie Deng**[2], **Tianqi Hou**[3], **Robert C. Qiu**[1], **Zenan Ling**[1*]
[1]School of EIC, Huazhong University of Science and Technology
[2]School of Computer Science, Shanghai Jiao Tong University [3]Huawei

Abstract

Consistency Models (CMs) have shown promise for efficient one-step generation. However, most existing CMs rely on manually designed discretization schemes, which can cause repeated adjustments for different noise schedules and datasets. To address this, we propose a unified framework for the automatic and adaptive discretization of CMs, formulating it as an optimization problem with respect to the discretization step. Concretely, during the consistency training process, we propose using local consistency as the optimization objective to ensure trainability by avoiding excessive discretization, and taking global consistency as a constraint to ensure stability by controlling the denoising error in the training target. We establish the trade-off between local and global consistency with a Lagrange multiplier. Building on this framework, we achieve adaptive discretization for CMs using the Gauss-Newton method. We refer to our approach as ADCMs. Experiments demonstrate that ADCMs significantly improve the training efficiency of CMs, achieving superior generative performance with minimal training overhead on both CIFAR-10 and ImageNet. Moreover, ADCMs exhibit strong adaptability to more advanced DM variants. Code is available at `https://github.com/rainstonee/ADCM`.

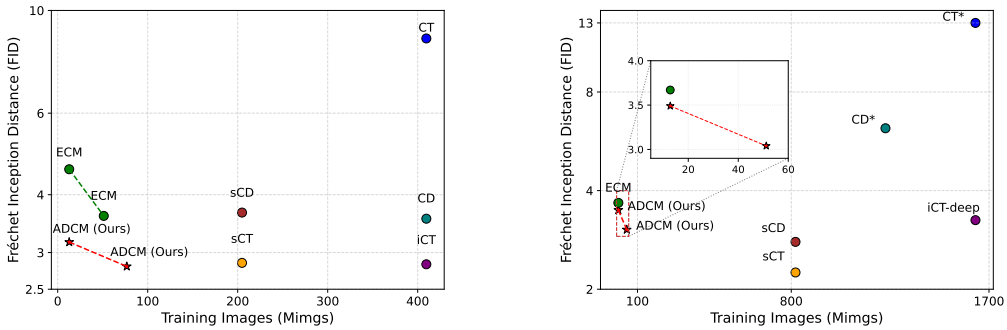
1 Introduction

Diffusion Models (DMs) [34, 9, 38, 18, 20] have achieved remarkable accomplishments in the field of data generation, including images [4, 29, 30], videos [10, 2, 41], audio [15, 28, 19] and 3D contents [40, 27, 22]. However, DMs require numerous iterations to achieve high-quality generation, significantly slowing sampling speed and making it resource-intensive. Recently, many fast-sampling methods for DMs have been proposed, including training-free methods [35, 14, 24, 48] and distillation-based approaches [25, 31, 46, 6, 26, 42, 45, 32]. However, these methods often sacrifice quality for faster sampling or incur substantial training overhead, which limits their practical application.

Consistency Models (CMs) [37, 7] offer significant advantages in addressing these challenges. CMs sample trajectories from the PF-ODE of DMs and aim to map each point on these trajectories to their corresponding endpoint. Through this approach, CMs achieve single-step generation while preserving the advantage of DMs, which improve generation quality by performing more iterations. CMs achieve the mapping to the endpoint by minimizing the distance between adjacent trajectory points. We refer to the selection of adjacent trajectory points as the *discretization* for CMs. Previous works [37, 36, 7, 23, 21] have shown that discretization is crucial for CMs' training. It determines the trainability and stability of CMs: poor trainability can impair final performance, while instability during training may slow convergence or even lead to divergence. A suboptimal discretization strategy may lead to an imbalance between trainability and stability [7]. It may also cause CMs to overly focus on training within a specific time interval, leading to a loss of consistency [17]. To mitigate these challenges and ensure a balanced training process, most existing works adopt empirical discretization strategies, which require manual adjustments based on different noise schedules and datasets [7].

*Corresponding Author: Zenan Ling (lingzenan@hust.edu.cn).

39th Conference on Neural Information Processing Systems (NeurIPS 2025).



(a) CIFAR-10 (b) ImageNet 64×64

Figure 1: ADCMs significantly improve the **training efficiency** on both (a) unconditional CIFAR-10 and (b) class-conditional ImageNet 64×64. ADCMs achieve superior generation quality with only a minimal amount of training data. $*$ indicates a smaller model.

Our fundamental goal is to adaptively determine the discretization strategy for CMs' training[2], considering both trainability and stability, thereby improving the training efficiency and final performance of CMs. First, we propose that the discretization step should minimize the optimization objective of CMs, i.e., *local consistency*, to ensure their trainability. Second, the discretization step controls the denoising error in the training target of CMs, which affects the *global consistency*. Excessive denoising error can lead to instability in CMs' training, thereby degrading the efficiency of CMs. Hence, we introduce global consistency as a constraint to ensure stability. To adaptively balance trainability and stability, we formulate local and global consistency as a constrained optimization problem and relax it via the Lagrangian method, introducing a Lagrange multiplier to express the trade-off between them. To achieve adaptive discretization, we propose using the Gauss-Newton method to obtain an analytical solution to the optimization problem. We refer to our method as Adaptive Discretization for Consistency Models (ADCMs). Our analysis reveals that ADCMs adaptively adjust discretization steps by jointly considering local and global consistency, thus achieving a balanced trade-off between trainability and stability.

In our experiments, ADCMs significantly improve the training efficiency and final performance of CMs. On unconditional CIFAR-10, as shown in Figure 1a, ADCMs exhibit outstanding training efficiency. We achieve a 1-step FID of 3.16 with only a training budget of 12.8M images. In contrast, ECM [7], the most efficient CM from previous work, requires 51.2M images to reach a FID of 3.60. Moreover, we attain a 1-step FID of 2.80 using only 76.8M images, outperforming iCT [36], which requires 409.6M images to achieve comparable performance. On class-conditional ImageNet 64×64, as shown in Figure 1b, ADCMs significantly reduce the training overhead under the same model size. ADCMs achieve a 1-step FID of 3.49 with a training budget of only 12.8M images. When the training budget increases to 51.2M images, ADCMs achieve a competitive 1-step FID of 3.04.

Contributions. Our contributions are summarized as follows.

- We provide a unified framework for the discretization for CMs. Starting from local consistency and global consistency, we investigate the impact of discretization steps on the trainability and stability of CMs. Guided by these two principles, we formulate a constrained optimization problem for selecting the discretization step. Previous discretization methods can be regarded as special cases of our framework.
- Based on this framework, we propose Adaptive Discretization for Consistency Models (ADCMs). First, we relax the optimization problem using the Lagrangian method and establish a trade-off between the trainability and stability of CMs through the Lagrange multiplier. Then, we employ the Gauss-Newton method to derive an analytical solution to the optimization problem, enabling adaptive discretization steps that effectively balance

[2]In particular, we focus on Consistency Training (CT) [37] over Consistency Distillation (CD) due to its superior empirical performance and its ability to bypass numerical solvers by directly leveraging training data for unbiased score estimation.

local and global consistency. Additionally, we introduce an adaptive loss function to further improve performance.

- Our experiments demonstrate that ADCMs significantly improve the training efficiency, while achieving competitive performance in one-step generation. On CIFAR-10, ADCMs achieve superior results using less than 25% of the training budget compared to previous works. On ImageNet, ADCMs also demonstrate strong performance with minimal training overhead. Furthermore, ADCMs adapt to advanced variants of DMs such as Flow Matching without manual adjustments.

1.1 Related Works

Consistency Models. Consistency Models were first proposed by [37], achieving the distillation of Diffusion Models by mapping any point on the PF-ODE trajectory to the endpoint of the trajectory. To accomplish this, it proposed sampling adjacent points on the trajectory and enforcing that the output near the noise end approximates the output near the data end. It divided CMs into two categories: Consistency Distillation (CD) and Consistency Training (CT), corresponding to sampling trajectory points using a pretrained DM and the forward diffusion process, respectively. iCT [36] explored the potential of CT, as it does not require a pretrained DM to sample trajectory points, thus supporting training from scratch. ECM [7] discovered that initializing the CM with a pretrained DM can effectively accelerate its training speed. TCM [17] discovered that CMs struggle to map the entire trajectory using a single model. Therefore, it proposed a two-stage approach for CMs, enabling CMs to focus on learning tasks from different time intervals separately. CTM [13] and Shortcut Models [5] aimed to make CMs capable of mapping to any point on the trajectory, not just the endpoint, by introducing an additional time condition to assist the model's learning.

Discretization for CMs. The training of CMs fundamentally relies on the selection of adjacent trajectory points, a process we refer to as the discretization for CMs. Various discretization strategies have been explored in previous works. iCT [36] proposed segmenting time based on the sampling steps of DMs [11]. ECM [7] introduced a decoupled approach, employing two functions: one to determine the overall magnitude of discretization steps and another to regulate their distribution over time. Both iCT and ECM adopt exponentially decreasing time steps to enhance the stability of CMs' training. Alternatively, CCM [21] introduced an adaptive discretization scheme by iteratively solving for the discretization step based on a Peak Signal-to-Noise Ratio (PSNR) threshold, ensuring a more balanced training across different times. sCM [23] formulated an "infinite" discretization approach, where adjacent trajectory points become infinitesimally close, transforming their distance into the first-order time derivative. However, sCM observed that this discretization scheme suffers from stability issues and proposed modifications to both the noise schedule and the neural network architecture, among other refinements, to ensure stable training.

2 Preliminaries

2.1 Diffusion Models

Given a dataset with an underlying distribution p_{data}, DMs generate samples by learning to reverse a noising process that progressively adds random Gaussian noise to the data, eventually transforming it into pure noise. Specifically, for a data sample $\boldsymbol{x}_0 \sim p_{\mathrm{data}}$ and a noise sample $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$, the diffusion process is defined as:

$$\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_0 + \beta_t \boldsymbol{z}$$

where $t \in [\epsilon, T]$, and ϵ is a small value used to prevent numerical errors. [38] proposes that the diffusion process can be modeled as a forward SDE, which is then denoised step-by-step using the corresponding probability flow ODE (PF-ODE). DMs utilize a time-dependent neural network (NN) to predict the unknown $\boldsymbol{x}_0$ in the PF-ODE. The optimization objective of DMs is given by:

$$\min_{\theta} \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{z}, t}[w(t) \cdot \|f_\theta(\boldsymbol{x}_t) - \boldsymbol{x}_0\|_2^2] \tag{1}$$

where $f_\theta(\boldsymbol{x}_t) = c_{\text{skip}}(t)\boldsymbol{x}_t + c_{\text{out}}(t)F_\theta\big(c_{\text{in}}(t)\boldsymbol{x}_t, c_{\text{noise}}(t)\big)$, $w(t)$ is a weighting function and F_θ is an NN with parameters θ. We write $f(\boldsymbol{x}_t, t)$ as $f(\boldsymbol{x}_t)$ for simplicity. Most DMs, including DDPM [9], EDM [11], and Flow Matching [18], have training objectives that can be equivalently expressed as Eq. (1) through the design of precondition $c_{\text{skip}}(t)$ and $c_{\text{out}}(t)$.

2.2 Consistency Models

CMs [37, 23] aim to map any point $\boldsymbol{x}_t$ on the PF-ODE trajectory to the corresponding data $\boldsymbol{x}_0$, i.e., $f_\theta(\boldsymbol{x}_t, t) = \boldsymbol{x}_0$. To achieve this, (1) at $t = 0$, CMs require that f_θ satisfy the boundary condition $f_\theta(\boldsymbol{x}_\epsilon, \epsilon) \equiv \boldsymbol{x}_0$, which implies that $c_{\text{skip}}(\epsilon) = 1$ and $c_{\text{out}}(\epsilon) = 0$; (2) for $t > 0$, CMs are trained to produce consistent outputs for any two adjacent points on the PF-ODE trajectory. Specifically, the optimization objective of CMs is given by:

$$\min_\theta \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{z}, t}[w(t) \cdot \|f_\theta(\boldsymbol{x}_t) - f_{\theta^-}(\boldsymbol{x}_{t-\Delta t})\|_2^2] \tag{2}$$

where θ^- stands for $\text{stopgrad}(\theta)$ and Δt is the time interval that defines the adjacent time step corresponding to a given time t, which in turn determines the training target on the PF-ODE trajectory. When retrieving the adjacent point $\boldsymbol{x}_{t-\Delta t}$, we adopt Consistency Training (CT) paradigm [37] which enables unbiased estimation of the score function, expressed as $\frac{\boldsymbol{x}_t - \alpha_t \boldsymbol{x}_0}{\beta_t^2}$. This approach eliminates the numerical errors introduced by solvers required for Consistency Distillation (CD).

The choice of Δt, referred to as the *discretization* of CMs, plays a crucial role in their training [37, 21, 23]. Previous works have proposed various discretization strategies, which can be categorized into two types, as outlined below.

Discrete-CMs. When $\Delta t > 0$ is not infinitesimally small, CMs fall to discrete-CMs. Discrete-CMs require careful planning of the discretization schedule. iCT [36] divides the time interval $[\epsilon, T]$ into multiple segments using the sampling time steps in DMs [11], i.e., $\mathbb{T} = \{t_0, \ldots, t_N\}$ where $t_0 = \epsilon$ and $t_N = T$, and samples adjacent time points within $\mathbb{T}$ as t and $t - \Delta t$. iCT proposes that the discretization of CMs requires meticulous planning and designs time steps that decrease progressively during training. ECM [7] also adopts dynamic time step scheduling. Unlike iCT, ECM samples t in continuous time and then maps the corresponding time step through a manually designed function. CCM [21] points out that the discretization step size affects the training difficulty of CMs at different times. CCM proposes setting a PSNR threshold for CMs and solving the PF-ODE iteratively until the selected time steps satisfy this threshold.

Continuous-CMs. Taking the limit $\Delta t \to 0$, CMs fall to continuous-CMs, which can be considered equivalent to "infinite" discretization. [37] proves that continuous-CMs can be optimized with:

$$\nabla_\theta \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{z}, t}\left[w(t) f_\theta^\top(\boldsymbol{x}_t) \frac{\mathrm{d} f_{\theta^-}(\boldsymbol{x}_t)}{\mathrm{d}t}\right] \tag{3}$$

which is a continuous version of Eq. (2). Continuous-CMs effectively avoid the discretization schedule of discrete-CMs. However, continuous-CMs often face significant instability challenges. sCM [23] addresses this instability for a specific DM with a specialized noise schedule, but it remains unclear that how to address the instability for continuous CMs under a more general setting.

Overall, the choice of discretization step Δt is still challenging. If Δt is too large, CMs struggle to learn meaningful information, while if it is too small, instability issues arise [23]. Additionally, how Δt varies *w.r.t.* t determines the training emphasis at different times [36, 7], and suboptimal strategies can lead the model to focus on specific time intervals, negatively impacting overall performance. Although various discretization strategies have been proposed, they often fail to identify the optimal Δt for each time step. On the one hand, existing discrete-CMs lack adaptive adjustment capabilities, requiring additional modeling and hyperparameter tuning for different noise schedules and datasets. On the other hand, continuous-CMs avoid discrete time steps by treating all time steps equally, but not all are equally important for effective training [36, 17]. This limits training efficiency.

3 Methodology

3.1 ADCMs: Adaptive Discretization for Consistency Models

In Section 2.2, we illustrate the importance of discretization in training CMs. In this study, our fundamental goal is to determine which discretization strategy is most beneficial for CMs' training, i.e., the discretization step Δt for a *given* time t. When we fix the NN's parameters $\theta^- = \text{stopgrad}(\theta)$ and time t, we aim to find an optimal Δt that contributes the most to the following training objective of CMs:

$$\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{z}}[\|f_{\theta^-}(\boldsymbol{x}_t) - f_{\theta^-}(\boldsymbol{x}_{t-\Delta t})\|_2^2]. \tag{4}$$

We define Eq. (4) as *local consistency* as it reflects the properties of CMs in a local interval. First, we need that the objective in Eq. (4) is trainable. To achieve this, we need to choose an appropriate Δt such that the objective is as small as possible, thereby satisfying local consistency, namely:

$$\min_{\Delta t} \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{z}} \left[\| f_{\theta^-}(\boldsymbol{x}_t) - f_{\theta^-}(\boldsymbol{x}_{t-\Delta t}) \|_2^2 \right]. \tag{5}$$

It can be observed that when $\Delta t = 0$, the local consistency in Eq. (5) is minimized. This implies that we need to choose Δt as small as possible. However, previous works [37, 7, 23] have shown that when Δt is too small, CMs face severe stability issues, which slows down convergence and may even lead to divergence. The underlying reason is that the practical training target, i.e., $f_{\theta^-}(\boldsymbol{x}_{t-\Delta t})$, fails to precisely denoise $\boldsymbol{x}_{t-\Delta t}$ to the ground-truth $\boldsymbol{x}_0$, leading to the global denoising error quantified as:

$$\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{z}} \left[\| f_{\theta^-}(\boldsymbol{x}_{t-\Delta t}) - \boldsymbol{x}_0 \|_2^2 \right]. \tag{6}$$

Remark 3.1. This denoising error is also an upper bound on the squared Wasserstein-2 distance between p_{data} and the data distribution generated by f_{θ^-} at time $t - \Delta t$. Moreover, this error can be regarded as a lower bound on the accumulated error from previous time steps, namely:

$$\sqrt{\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{z}} \left[\| f_{\theta^-}(\boldsymbol{x}_{t-\Delta t}) - \boldsymbol{x}_0 \|_2^2 \right]} \leq \sum_{i=1}^{k} \sqrt{\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{z}} \left[\| f_{\theta^-}(\boldsymbol{x}_{t_i}) - f_{\theta^-}(\boldsymbol{x}_{t_i-\Delta t_i}) \|_2^2 \right]},$$

where Δt_i is the discretization step corresponding to time t_i, satisfying $t_i - \Delta t_i = t_{i-1}$.

We define Eq. (6) as *global consistency* because it reflects the global properties of CMs. Excessive denoising error will cause CMs to optimize in the wrong direction, which leads to instability in CMs' training. Therefore, we propose that when selecting the discretization step Δt, it is necessary to ensure that the denoising error is constrained, namely:

$$\text{Find} \quad \Delta t, \quad \text{s.t.} \quad \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{z}} \left[\| f_{\theta^-}(\boldsymbol{x}_{t-\Delta t}) - \boldsymbol{x}_0 \|_2^2 \right] \leq \delta \tag{7}$$

where δ is an upper bound that needs to be set manually. Clearly, when Δt takes the maximum value $t - \epsilon$, due to the boundary condition $f_\theta(\boldsymbol{x}_\epsilon, \epsilon) \equiv \boldsymbol{x}_0$, the constraint in Eq. (7) will be satisfied regardless of the value of δ. This implies that we need to choose the largest possible Δt.

Notably, the optimization objective in Eq. (5) and the constraint in Eq. (7) respectively impose opposite guidance for Δt. When Δt is extremely small, the local consistency error in Eq. (5) is minimized, making it easy for CMs to optimize. However, this may cause the constraint in Eq. (7) to exceed its upper bound. Conversely, when Δt is extremely large, the constraint in Eq. (7) will be easily satisfied, but it may cause the optimization objective in Eq. (5) to become too large and difficult to optimize. Therefore, we propose a constrained optimization objective to achieve a trade-off between Eq. (5) and Eq. (7), which is given by:

$$\min_{\Delta t} \quad \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{z}} \left[\| f_{\theta^-}(\boldsymbol{x}_t) - f_{\theta^-}(\boldsymbol{x}_{t-\Delta t}) \|_2^2 \right], \quad \text{s.t.} \quad \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{z}} \left[\| f_{\theta^-}(\boldsymbol{x}_{t-\Delta t}) - \boldsymbol{x}_0 \|_2^2 \right] \leq \delta. \tag{8}$$

We denote the optimization objective $\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{z}}[\| f_{\theta^-}(\boldsymbol{x}_t) - f_{\theta^-}(\boldsymbol{x}_{t-\Delta t}) \|_2^2]$ as $\mathcal{L}_{\text{local}}$, as it focuses on the local consistency information of CMs and controls the local consistency error for CMs. Therefore, minimizing $\mathcal{L}_{\text{local}}$ effectively improves the effectiveness of CMs. We denote the constraint $\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{z}}[\| f_{\theta^-}(\boldsymbol{x}_{t-\Delta t}) - \boldsymbol{x}_0 \|_2^2]$ as $\mathcal{L}_{\text{global}}$ as it focuses on the global consistency and controls the denoising error in the training objective. Consequently, $\mathcal{L}_{\text{global}}$ helps CMs effectively eliminate denoising error, find accurate optimization targets, and thus improve training stability and efficiency. Our goal is to ensure both the global consistency and the local consistency simultaneously, enabling an adaptive adjustment of CMs' discretization. However, directly optimizing the constrained optimization problem in Eq. (8) is challenging. To address this, we apply the Lagrange multiplier method to relax the problem, yielding the following formulation:

$$\Delta t^* = \arg\min_{\Delta t} \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{z}}[\mathcal{L}_{\text{local}}(t, \Delta t) + \lambda \mathcal{L}_{\text{global}}(t, \Delta t)]. \tag{9}$$

Here, the Lagrange multiplier λ acts as a weighting factor balancing the local consistency and the global consistency of CMs. We aim for λ to be a constant independent of t, ensuring that the focus on trainability and stability remains consistent across different time scales. Typically, we set $\lambda \ll 1$, as ensuring whether CMs are trainable is of greater importance compared to their stability. We refer to our approach as Adaptive Discretization for Consistency Models (ADCMs), as shown in Figure 2. We find that previous discretization strategies can be unified in ADCMs. We summarize this as follows.

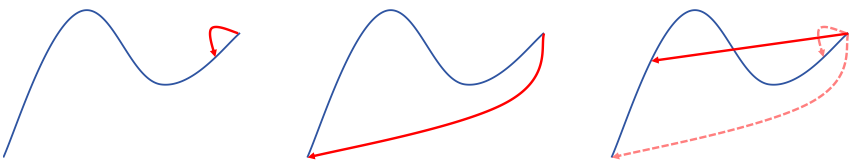


Figure 2: Discretization strategies of different models. CMs consider only local consistency during discretization, while DMs consider only global consistency. ADCMs balance local and global consistency and adaptively adjust the discretization step size based on the information from both.

Remark 3.2. DMs, discrete-CMs and continuous-CMs can be viewed as special cases of Eq. (9). Specifically,

- DMs [11, 18]: Choose the maximum optimization step $\Delta t = t - \epsilon$. this corresponds to set $\lambda \to \infty$ in our framework.
- Discrete-CMs:
  - CM [37], iCT [36], ECM [7]: These methods consider the smoother trajectory changes near the noise end and empirically choose a discretization step size that monotonically increases *w.r.t.* t. This is equivalent to estimating Eq. (9) empirically in our framework.
  - CCM [21]: CCM ensures that for all $\boldsymbol{x}_0, \boldsymbol{z}$, $\mathcal{L}_e$ is always less than a certain threshold δ. Since an analytical solution cannot be obtained directly, CCM requires iterative solving for all $\boldsymbol{x}_0, \boldsymbol{z}, t$. This is equivalent to making $\mathcal{L}_{\text{local}}$ a constant in our framework.
- Continuous-CMs [23]: Choose the minimum optimization step $\Delta t \to 0$. This is equivalent to set $\lambda = 0$ in our framework.

Analytical Solution. To achieve an adaptive solution for the discretization step, we propose using the Gauss-Newton method to directly derive an analytical solution to the optimization problem in Eq. (9). Since we assign a higher weight to local consistency in the objective, we approximate $f_{\theta^-}(\boldsymbol{x}_{t-\Delta t})$ using its first-order Taylor expansion, which is:

$$f_{\theta^-}(\boldsymbol{x}_{t-\Delta t}) \approx f_{\theta^-}(\boldsymbol{x}_t) - \boldsymbol{v}\Delta t, \quad \boldsymbol{v} = \nabla_{\boldsymbol{x}_t} f_{\theta^-} \cdot \frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} + \partial_t f_{\theta^-}$$

where $\boldsymbol{v}$ can be efficiently computed via the Jacobian-vector product (JVP) for $f_{\theta^-}(\cdot, \cdot)$, evaluated at input vector $(\boldsymbol{x}_t, t)$ and tangent vector $(\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t}, 1)$, following the method in [23]. Under this approximation, the optimization problem is transformed into a least-squares problem, whose optimal solution is given by:

$$\Delta t^* = \frac{\lambda}{1+\lambda} \frac{\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{z}}[\boldsymbol{v}^\top (f_{\theta^-}(\boldsymbol{x}_t) - \boldsymbol{x}_0)]}{\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{z}}[\boldsymbol{v}^\top \boldsymbol{v}]}. \tag{10}$$

From the expression of the discretization step, we have the following observations:

1. The optimal discretization step is inversely proportional to the magnitude of the Jacobian. This indicates that the output of the current network may vary significantly, and $\mathcal{L}_{\text{local}}$ could be very large. Therefore, a smaller step size is required to ensure effectiveness.
2. The optimal discretization step is proportional to $\|f_{\theta^-}(\boldsymbol{x}_t) - \boldsymbol{x}_0\|_2$, which is an effective estimate of $\mathcal{L}_{\text{global}}$. This indicates that the denoising error may be very large at this time, and therefore, a larger step size is required to ensure stability.
3. The optimal discretization step is proportional to the linear correlation between $\boldsymbol{v}$ and $f_{\theta^-}(\boldsymbol{x}_t) - \boldsymbol{x}_0$. This indicates that when $\boldsymbol{v}$ and $f_{\theta^-}(\boldsymbol{x}_t) - \boldsymbol{x}_0$ tend toward linearity, the direction of local optimization aligns more closely with the direction of global optimization, allowing for the use of a larger step size.

Algorithm 1 Adaptive Discretization for Consistency Models

Input: dataset $\mathcal{D}$, diffusion parameter α_t and β_t, time range $[\epsilon, T]$, network parameter θ, weighting factor λ, update frequency m, batch size b
$\theta^- \leftarrow \theta$
repeat
  Initialize an empty set $\mathbb{T}$ and $t \leftarrow T$
  repeat
    Append t to $\mathbb{T}$
    Sample mini-batch $\boldsymbol{x}_0 \sim \mathcal{D}$, $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$
    $\boldsymbol{x}_t \leftarrow \alpha_t \boldsymbol{x}_0 + \beta_t \boldsymbol{z}$
    Calculate Δt^* through Eq. (10)
    $t \leftarrow t - \Delta t^*$
  until $t \leq \epsilon$
  Append ϵ to $\mathbb{T}$
  for $k = 1$ **to** m **do**
    Sample mini-batch $\boldsymbol{x}_0 \sim \mathcal{D}$, $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$ and adjacent time points $t, t - \Delta t^* \sim \mathbb{T}$
    $\boldsymbol{x}_t \leftarrow \alpha_t \boldsymbol{x}_0 + \beta_t \boldsymbol{z}$, $\boldsymbol{x}_{t-\Delta t^*} \leftarrow \alpha_{t-\Delta t^*} \boldsymbol{x}_0 + \beta_{t-\Delta t^*} \boldsymbol{z}$
    Calculate loss $\mathcal{L}$ through Eq. (11)
    Update θ using $\mathcal{L}$
  end for
until Convergence

The above analysis demonstrates that the proposed discretization step can be adaptively adjusted based on the current state of the NN, considering both $\mathcal{L}_{\text{global}}$ and $\mathcal{L}_{\text{local}}$. As a result, we achieve an adaptive balance between the stability and trainability of CMs at different times through Eq. (10). Starting from $t = T$, we iteratively solve the optimization problem to derive the adaptively optimal time segmentation $\mathbb{T} = \{t_1^*, \ldots, t_N^*\}$.

3.2 Putting ADCMs into Practice: Further Training Strategies

Adaptive Weighting Function. Through the analysis in Section 3.1, we know that $\mathcal{L}_{\text{global}}$ fundamentally determines the training stability of CMs at the current time t. However, during the training of CMs, the NN only optimizes for $\mathcal{L}_{\text{local}}$. Therefore, to further balance the impact of $\mathcal{L}_{\text{global}}$ over time, we propose the following adaptive weighting function:

$$w(t) = \frac{1}{\mathcal{L}_{\text{global}}} = \frac{1}{\|f_{\theta^-}(\boldsymbol{x}_{t-\Delta t}) - \boldsymbol{x}_0\|_2^2}.$$

When $\mathcal{L}_{\text{global}}$ is very large, the training of CMs will suffer from instability. Therefore, a smaller weighting is needed. On the other hand, when $\mathcal{L}_{\text{global}}$ is small, the CMs' training objective aligns closely with the true target, and thus a larger weighting is required.

Adaptive Distance Metric. Previous works [7, 36] have shown that compared to the squared L_2 metric, Pseudo-Huber metric can effectively reduce training variance, which is given by:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + c^2} - c$$

where c is a constant. ADCMs also use Pseudo-Huber metric for training. At the same time, in order to ensure the consistency of the distance function, we have similarly modified the adaptive weighting function. The overall loss function of ADCMs can be expressed as:

$$\min_{\theta} \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{z}, t} \left[\frac{\sqrt{\|f_\theta(\boldsymbol{x}_t) - f_{\theta^-}(\boldsymbol{x}_{t-\Delta t^*})\|_2^2 + c^2} - c}{\sqrt{\|f_{\theta^-}(\boldsymbol{x}_{t-\Delta t^*}) - \boldsymbol{x}_0\|_2^2 + c^2} - c} \right] \tag{11}$$

where Δt^* is obtained with Eq. (10). See Appendix B for more discussion on loss function design.

Putting It Together. We alternately optimize the time segmentation $\mathbb{T}$ and the NN's parameters θ during training. We typically update $\mathbb{T}$ after updating θ for $m = 25000$ times, as the changes in the

Table 1: Sample quality on unconditional CIFAR-10 and class-conditional ImageNet 64×64. $*$ indicates additional training costs.

Method	CIFAR-10		ImageNet 64×64	
	NFE (↓)	FID (↓)	NFE (↓)	FID (↓)
Diffusion Models				
DDPM [9]	1000	3.17	250	11.0
EDM [11]	35	1.97	511	1.36
DPM-Solver [24]	10	4.70	20	3.42
1-Rectified Flow [20]	127	2.58	–	–
ADM [4]	–	–	250	2.07
EDM2-S [12]	–	–	63	1.58
EDM2-XL [12]	–	–	63	1.33
Joint Training				
StyleGAN-XL [33]	1	1.52	1	1.52
SiD [47]	1	1.92	1	1.52
CTM [13]	1	1.87	1	1.92
CCM [21]	1	1.64	1	2.18
Consistency-FM [43]	2	1.69	2	1.62
DMD2 [44]	–	–	1	1.28
Diffusion Distillation				
DFNO (LPIPS) [46]	1	3.78	1	7.83
PID (LPIPS) [39]	1	3.92	1	9.49
TRACT [1]	1	3.78	1	7.43
PD [31]	1	8.34	1	10.70
2-Rectified Flow [20]	1	4.85	–	–
Consistency Models				
CD (LPIPS) [37]	1	3.55*	1	6.20*
CT [37]	1	8.70	1	13.0
iCT [36]	1	2.83	1	4.02
iCT-deep [36]	1	2.51*	1	3.25
ECM [7]	1	3.60	1	2.49*
TCM [17]	1	2.46*	1	2.20*
sCD [23]	1	3.66	1	2.44*
sCT [23]	1	2.85	1	2.04*
ADCM (ours)	1	2.80	1	3.04

Table 2: Training efficiency on CIFAR-10.

Unconditional CIFAR-10		
Method	**Training Budget (Mimgs)**	**1-Step FID (↓)**
CD (LPIPS)	409.6	3.55
iCT	409.6	2.83
sCT (TrigFlow)	204.8	2.85
sCT (VE)	51.2	4.18
ECM	12.8	4.54
ECM	51.2	3.60
ADCM (Ours)	12.8	3.16
ADCM (Ours)	76.8	**2.80**

Table 3: Training efficiency on ImageNet 64×64.

Class-Conditional ImageNet 64×64			
Method	**Model Size**	**Training Budget (Mimgs)**	**1-Step FID (↓)**
CD (LPIPS)	1×	1228.8	6.20
iCT	1×	1638.4	4.02
iCT-deep	2×	1638.4	3.25
sCT (TrigFlow)	2×	819.2	2.25
ECM	1×	12.8	5.51
ECM	2×	12.8	3.67
ADCM (Ours)	1×	12.8	5.12
ADCM (Ours)	1×	25.6	4.65
ADCM (Ours)	1×	51.2	4.23
ADCM (Ours)	2×	12.8	3.49
ADCM (Ours)	2×	25.6	3.28
ADCM (Ours)	2×	51.2	3.04

NN are relatively slow. Before training the network, we fix its parameters and perform simulation-based optimization starting from $t = T$. We iteratively update t using Eq. (10) until $t = \epsilon$, recording the optimization process as $\mathbb{T} = \{t_1^*, \ldots, t_N^*\}$. We observe that during the optimization process, the expectation in Eq. (10) is well approximated using a single mini-batch. This is because we do not require precise step sizes, only the trend of their change over time t. Subsequently, we fix the time segmentation and optimize the NN. The detailed process is illustrated in Algorithm 1.

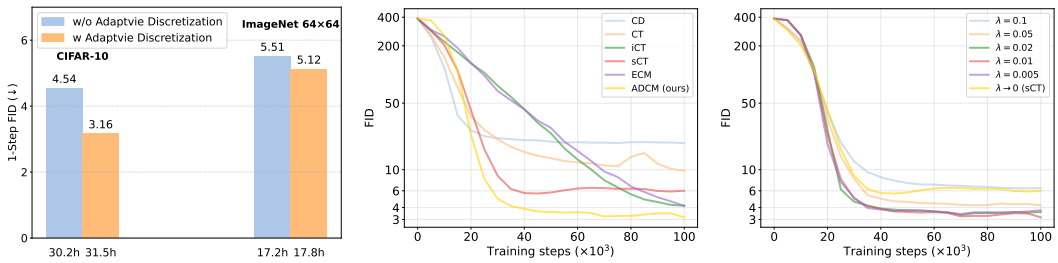
4 Experiments

To validate the effectiveness of ADCMs, we perform unconditional and class-conditional generation experiments on CIFAR-10 [16] and ImageNet 64×64 [3], respectively. For CIFAR-10, we initialize CMs with pretrained DM from [11]. For ImageNet 64×64, we adopt the pretrained DM from [12]. If not otherwise specified, our experiments are conducted under VE SDE [38] settings. We evaluate the sample quality using FID [8] and measure the generation speed using the number of function evaluations (NFEs).

We compare ADCMs with different generative models, as shown in Table 1. FIDs with $*$ indicate that they have additional training costs compared to other CMs, such as a larger model or an auxiliary model used during training. Experiments show that ADCMs achieve high-quality single-step generation without additional training costs. See Appendix C for multi-step generation results.

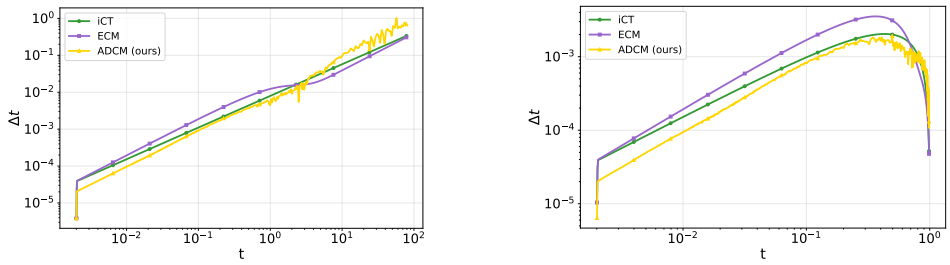
4.1 Efficiency of ADCMs

We evaluate the training efficiency of ADCMs on both unconditional CIFAR-10 and class-conditional ImageNet 64×64. For CIFAR-10, we use a unified model size and measure computational cost by the total number of training images. For ImageNet 64×64, both model size and training budgets are taken into consideration. TCM [17] is excluded from the comparison since its two-stage strategy introduces significant training overhead. For a fair comparison, we reproduce some baseline results, as detailed in Appendix A.3.



(a) (b) (c)

Figure 3: (a) Training time cost of ADCMs. (b) Training dynamics of different discretization methods. Compared to other CMs' approaches, ADCMs have a faster convergence rate and better final performance. (c) Training dynamics for different λ. A large λ improves stability but hurts final performance, while a too-small λ reduces stability and hinders convergence.



(a) EDM (VE) (b) Flow Matching

Figure 4: Adaptive Discretization on (a) EDM (VE) and (b) Flow Matching.

Data Efficiency. On unconditional CIFAR-10, as shown in Table 2, ADCMs achieve high-quality one-step generation results with a training budget of only 12.8M images. Compared with ECM [7], the most efficient CM to date, ADCMs achieve better generation quality with only 25% of its training budget. Moreover, ADCMs surpass all previous CMs in one-step generation performance with only 76.8M training images. On class-conditional ImageNet 64×64, as shown in Table 3, ADCMs significantly reduce the training budgets of CMs. Compared to the most efficient ECM [7], ADCMs can achieve a better 1-step FID with the same model size and training budget. Moreover, ADCMs exhibit notable improvements as both the model size and training budget increase. With a $2\times$ model size, ADCMs achieve a 1-step FID of 3.49 with a training budget of only 12.8M images. Remarkably, ADCMs are able to surpass iCT-deep [36] with only 3% of its training budget.

Computational Efficiency. We first compare the training time cost of ADCMs with other CMs, as shown in Figure 3a. It can be observed that ADCMs introduce only about 4% additional time cost under the same training epochs while improving the final performance. We also explore the convergence speed of ADCMs on unconditional CIFAR-10 with different CM approaches, as shown in Figure 3b. It can be observed that ADCMs converge significantly faster than other CM approaches, while also achieving better final performance.

4.2 More Results

Adaptive Discretization Step of ADCMs. We explore the relationship between the adaptive discretization step Δt of ADCMs and time t under different noise schedules, and compare it with existing discretization methods. We modify the discretization strategies of iCT and ECM under Flow Matching setting to be functions of SNR in order to enhance their performance. The results are shown in Figure 4. ADCMs are able to adaptively learn discretization strategies that are similar in trend to empirical ones without manual adjustments. In addition, compared to other discretization schemes, ADCMs adopt finer discretization at smaller t and coarser discretization at larger t. As a result, ADCMs place greater emphasis on time intervals closer to the data during training, which aligns with empirical practices in both DMs and CMs [11, 36, 7].

Table 4: Generalization to Flow Matching. * indicates additional training costs.

Method	NFE (↓)	FID (↓)
1-Rectified Flow [20]	1	378
2-Rectified Flow* [20]	1	4.85
CCM* [21] (w GAN)	1	1.64
Consistency-FM (w/o GAN) [43]	2	5.34
ECM [7]	1	5.82
sCT [23]	1	88.52
ADCM (Ours)	1	5.14

Table 5: Scalability to ImageNet 512×512.

Class-Conditional ImageNet 512×512			
Method	Model Size	Training Budget (Mimgs)	1-Step FID (↓)
sCT	$1\times$	204.8	10.13
sCT	$2\times$	204.8	5.84
ECM	$1\times$	6.4	25.69
ECM	$2\times$	6.4	13.55
ADCM (Ours)	$1\times$	6.4	23.12
ADCM (Ours)	$2\times$	6.4	10.53

λ as a Trade-off between Stability and Effectiveness. We control the trade-off between the trainability and stability of ADCMs through the Lagrange multiplier λ according to Eq. (9). We perform an ablation study on λ by examining the training dynamics of ADCMs on unconditional CIFAR-10, as shown in Figure 3c. We find that when λ is small, i.e., more emphasis is placed on $\mathcal{L}_{\text{global}}$, CMs converge quickly, but the final generation quality is relatively poor. When λ is large, i.e., more emphasis is placed on $\mathcal{L}_{\text{local}}$, CMs become more unstable, making them difficult to converge and reach the optimal solution. Ablation study on loss function are deferred to Appendix B.

ADCMs Adapt to Different Variants of DMs. We conduct experiments on Flow Matching [18, 20], an advanced variant of DMs. We initialize CMs with pretrained DMs from [20] and compare ADCMs with other Flow Matching-based distillation methods. Additionally, we conduct experiments on ECM and sCT, two state-of-the-art CMs. All CMs are trained under a training budget of 12.8M images. As shown in Table 4, ADCMs achieve superior performance over other CMs without manual adjustments, which demonstrates the strong adaptability.

Scalability to High-Resolution Images. To further assess the scalability of ADCM, we conduct experiments on ImageNet 512×512. We adopt EDM2 [12] as the base latent diffusion model, which employs SD-VAE for image encoding and decoding. We compare ADCM with sCT [23] and ECM [7], the two most efficient prior CMs. The detailed results are reported in Table 5. It can be observed that ADCM scales effectively to large-scale datasets. As the model size increase, its performance improves substantially. Moreover, ADCM consistently outperforms ECM under the same training cost, further demonstrating its empirical effectiveness and training efficiency.

5 Conclusion

In this paper, we propose ADCMs, a unified framework for adaptive discretization in CMs. By formulating discretization as an optimization problem, we introduce local consistency as the optimization objective and global consistency as a constraint, establishing a trade-off using the Lagrange multiplier. Leveraging the Gauss-Newton method, ADCMs enable adaptive discretization, improving both trainability and stability. Experimental results show that ADCMs significantly improve training efficiency and final performance of CMs on different datasets while demonstrating strong adaptability to different variants of DMs.

Acknowledgments and Disclosure of Funding

Z. Ling is partially supported by the National Natural Science Foundation of China (via NSFC-62406119), the Natural Science Foundation of Hubei Province (2024AFB074), and the Guangdong Provincial Key Laboratory of Mathematical Foundations for Artificial Intelligence (2023B1212010001). Z. Deng is partially supported by the National Natural Science Foundation of China (via NSFC-92470118 and NSFC-62306176) and the Natural Science Foundation of Shanghai (23ZR1428700). R. C. Qiu is partially supported in part by the National Natural Science Foundation of China (via NSFC-12141107), the Key Research and Development Program of Wuhan (2024050702030100), and the Key Research and Development Program of Guangxi (GuiKe-AB21196034).

References

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

[2] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[4] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

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

[6] Zhengyang Geng, Ashwini Pokle, and J Zico Kolter. One-step diffusion distillation via deep equilibrium models. *Advances in Neural Information Processing Systems*, 36, 2024.

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

[8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

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

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

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

[12] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24174–24184, 2024.

[13] Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *The Twelfth International Conference on Learning Representations*, 2024.

[14] Zhifeng Kong and Wei Ping. On fast sampling of diffusion probabilistic models. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021.

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

[16] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[17] Sangyun Lee, Yilun Xu, Tomas Geffner, Giulia Fanti, Karsten Kreis, Arash Vahdat, and Weili Nie. Truncated consistency models. *arXiv preprint arXiv:2410.14895*, 2024.

[18] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[19] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. In *International Conference on Machine Learning*, pages 21450–21474. PMLR, 2023.

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

[21] Yunpeng Liu, Boxiao Liu, Yi Zhang, Xingzhong Hou, Guanglu Song, Yu Liu, and Haihang You. See further when clear: Curriculum consistency model. *arXiv preprint arXiv:2412.06295*, 2024.

[22] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9970–9980, 2024.

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

[24] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun 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.

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

[26] Thuan Hoang Nguyen and Anh 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.

[27] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023.

[28] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. Grad-tts: A diffusion probabilistic model for text-to-speech. In *International Conference on Machine Learning*, pages 8599–8608. PMLR, 2021.

[29] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

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

[31] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

[32] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pages 87–103. Springer, 2025.

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

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

[35] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[36] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *The Twelfth International Conference on Learning Representations*, 2024.

[37] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pages 32211–32252. PMLR, 2023.

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

[39] Joshua Tian Jin Tee, Kang Zhang, Hee Suk Yoon, Dhananjaya Nagaraja Gowda, Chanwoo Kim, and Chang D Yoo. Physics informed distillation for diffusion models. *arXiv preprint arXiv:2411.08378*, 2024.

[40] Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, Karsten Kreis, et al. Lion: Latent point diffusion models for 3d shape generation. *Advances in Neural Information Processing Systems*, 35:10021–10039, 2022.

[41] Yaohui Wang, Xinyuan Chen, Xin Ma, Shangchen Zhou, Ziqi Huang, Yi Wang, Ceyuan Yang, Yinan He, Jiashuo Yu, Peiqing Yang, et al. Lavie: High-quality video generation with cascaded latent diffusion models. *International Journal of Computer Vision*, pages 1–20, 2024.

[42] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36, 2024.

[43] Ling Yang, Zixiang Zhang, Zhilong Zhang, Xingchao Liu, Minkai Xu, Wentao Zhang, Chenlin Meng, Stefano Ermon, and Bin Cui. Consistency flow matching: Defining straight flows with velocity consistency. *arXiv preprint arXiv:2407.02398*, 2024.

[44] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. *arXiv preprint arXiv:2405.14867*, 2024.

[45] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6613–6623, 2024.

[46] Hongkai Zheng, Weili Nie, Arash Vahdat, Kamyar Azizzadenesheli, and Anima Anandkumar. Fast sampling of diffusion models via operator learning. In *International conference on machine learning*, pages 42390–42402. PMLR, 2023.

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

[48] Zhenyu Zhou, Defang Chen, Can Wang, and Chun Chen. Fast ode-based sampling for diffusion models in around 5 steps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7777–7786, 2024.

A Experiments Details

A.1 Precondition

For VE-based ADCMs, we follow the parameterization of EDM. Specifically, we set:

$$c_{\text{skip}}(t) = \frac{\sigma_{\text{data}}^2}{\sigma_{\text{data}}^2 + t^2},\ c_{\text{out}}(t) = \frac{\sigma_{\text{data}} t}{\sqrt{\sigma_{\text{data}}^2 + t^2}},\ c_{\text{in}}(t) = \frac{1}{\sqrt{\sigma_{\text{data}}^2 + t^2}},\ c_{\text{noise}}(t) = \frac{1}{4}\log t.$$

For ADCMs on the Flow Matching setting, using EDM's precondition causes the model output to deviate from $\boldsymbol{x}_0$, which contradicts the objective of CMs to estimate $\boldsymbol{x}_0$. We use a pretrained model from rectified flow [20] whose output is:

$$F_\theta(\boldsymbol{x}_t, t) = \frac{\boldsymbol{x}_t - \boldsymbol{x}_0}{t},$$

which implies $c_{\text{in}}(t) = 1$ and $c_{\text{noise}}(t) = t$. To ensure the model's final output matches $\boldsymbol{x}_0$, we accordingly modify the preconditioning to:

$$c_{\text{skip}}(t) = 1, \quad c_{\text{out}}(t) = -t.$$

A.2 Hyperparameters

Batch Size and EMA. For unconditional CIFAR-10, we use a batch size of 128 with an EMA decay rate of 0.9999 for training budget of 12.8M images. We use a batch size of 1024 with an EMA decay rate of 0.99993 for training budget of 76.8M images. For class-conditional ImageNet 64×64, we set the batch size to 128, 256, and 512, corresponding to training budgets of 12.8M, 25.6M, and 51.2M images, respectively. We use Power function EMA for class-conditional ImageNet 64×64 following [12].

Time Sampling. For unconditional CIFAR-10, we follow previous works [36, 7, 23] and use a log-normal SNR distribution for time sampling, which can be expressed as:

$$\log(\text{SNR}(t)) \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2)$$

where $\text{SNR}(t) = \frac{\beta_t}{\alpha_t}$, $P_{\text{mean}} = -1.1$, $P_{\text{std}} = 2.0$. Since the time segment $\mathbb{T}$ is discrete, we also apply discretization to the sampling results following [36]. For class-conditional ImageNet 64×64, we sample uniformly within the time segment $\mathbb{T}$.

Lagrange Multiplier λ. For unconditional CIFAR-10, we set $\lambda = 0.01$. For class-conditional ImageNet 64×64, we find that starting with a small λ led to training instability on ImageNet 64×64. Therefore, we follow previous work [36, 7] and select $\lambda = 0.64$ for warm-up, gradually decreasing it to $\lambda = 0.01$. We summarize the hyperparameter settings in Table 6.

A.3 Baseline Reimplementation.

Some of the baseline results in this paper, including those in Figure 3b, Table 4 and sCM [23] under VE settings, are obtained from our own reproductions. Under the VE SDE setting, for a fair comparison, we initialize all neural networks using the pretrained DM provided by EDM [11]. We also adopt a consistent EMA decay rate of 0.9999 and a dropout probability of 30% (except for CD where dropout is set to 0, as dropout can lead to a decline in CD's performance). We do not make further modifications to other parameters. For sCM under the VE SDE and Flow Matching setting, we apply the advanced training techniques from [23], except for the network architecture changes, allowing sCM to utilize pretrained DMs. We do not use the adaptive weighting and tangent warmup techniques, as we find that they degrade the performance of sCM. For all baselines under the Flow Matching setting, we replace their original discretization scheme, time sampling, and weighting function—from being functions of time t to being functions of SNR. It is important to note that without manual adjustments, the performance of these baselines degrades significantly (e.g., the 1-step FID of ECM drops from 5.82 to 15.55). The implementation code is available in the supplementary material.

Table 6: Hyperparameter Settings

	Unconditional CIFAR-10	Class-conditional ImageNet 64×64	
Base model	EDM [11]	EDM2-S [12]	EDM2-M [12]
Model capacity (Mparams)	55.7	280.2	497.8
Model complexity (GFLOPS)	21.3	101.9	180.8
GPU types	RTX3090	RTX3090	A100
GPU memory	24G	24G	40G
Number of GPUs	1	8	4
Dropout probability	30%	40%	50%
Optimizer	RAdam	Adam	Adam
Learning rate schedule	fixed	square root	square root
Learning rate max	0.0001	0.001	0.0009
Pseudo-Huber c	0.03	0	0
Time sampling	log-normal SNR	uniform	uniform
P_{mean}	-1.1	-	-
P_{std}	2.0	-	-

B Ablation Study

We investigate the impact of adaptive loss function in ADCMs, including the choice of weighting function and distance metric. We perform an ablation study on unconditional CIFAR-10 under the same training budget of 12.8M images.

Table 7: Ablation Study on Weighting Function.

Weighting Function	**1-Step FID (↓)**
1	5.70
$\frac{1}{t_i - t_{i-1}}$	4.09
$\frac{1}{t_i}$	3.84
Adaptive weighting (Ours)	**3.16**

Weighting Function. The choice of weighting function is crucial for training CMs. An inappropriate weighting function can lead to imbalanced optimization over time, ultimately degrading performance. We investigate the impact of different weighting functions on ADCMs. The detailed results are presented in Table 7. The results show that our designed adaptive weighting function can effectively enhance the generation capability of ADCMs. Notably, even without the loss function improvement, ADCMs still outperform ECM's 1-step FID (4.54). This demonstrates that the improvement of ADCMs mainly comes from our designed adaptive discretization strategy while our proposed adaptive weighting function further enhances the performance of ADCMs.

Distance Metric. We investigate the effect of different distance metrics on the performance of ADCMs. Following common practice in prior works [7, 36, 17], we adopt the Pseudo-Huber metric due to its robustness to outliers [36]. The Pseudo-Huber metric is defined as

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + c^2} - c,$$

which provides a smooth interpolation between the L_2 and squared L_2 metrics. Specifically, when $c = 0$, it reduces to the standard L_2 distance, while as $c \to \infty$, it approaches the squared L_2 distance. smoothly bridges the gap between the L_2 and squared L_2 metric. When $c = 0$, the Pseudo-Huber metric degenerates to the standard L_2 metric. When $c \to \infty$, it becomes equivalent to the squared L_2 metric. To address this phenomenon, we conduct experiments with different values of c, and the

Table 8: Ablation Study on Distance Metric.

Distance Metric	**1-Step FID (↓)**
L_2	3.54
Pseudo-Huber (c=0.0003)	3.44
Pseudo-Huber (c=0.003)	3.42
Pseudo-Huber (c=0.03)	**3.16**
Pseudo-Huber (c=0.3)	4.42
Pseudo-Huber (c=3)	5.23
Squared L_2	5.33

results are presented in Table 8. It can be observed that Pseudo-Huber metric smoothly interpolates between L_2 and squared L_2 through the control of the parameter c, thus achieving performance that surpasses both extremes. These results clearly demonstrate the significance of choosing Pseudo-Huber as our distance metric.

We also examine the impact of mismatched distance metrics between the original CM loss function and the weighting function. We fix the distance metric applied to the original CM loss function as Pseudo-Huber with $c = 0.03$, while applying different distance metrics in the weighting function. As shown in Table 9, a mismatched distance metric leads to degraded performance of ADCMs.

Table 9: Impact of Mismatched Distance Metric.

Distance Metric on Weighting Function	**1-Step FID (↓)**
Squared L_2	4.09
L_2	3.36
Pseudo-Huber (c=0.03)	**3.16**

C Two Step Generation

Compared to other distillation methods for DMs, CMs have the significant advantage of preserving the inherent characteristics of DMs, specifically, the ability to improve generation quality through multi-step sampling. We investigate the two-step generation performance of ADCMs on unconditional CIFAR-10, with results shown in Table 10. We set the intermediate $t = 0.420$. It can be observed that ADCMs not only maintain optimal single-step generation performance but also demonstrate strong two-step generation capabilities, second only to ECM [7], which is specifically designed for two-step generation.

Table 10: 2-step generation results on unconditional CIFAR-10. ADCMs achieve the best 1-step FID while also attaining the second-best 2-step FID.

Method	**1-Step FID (↓)**	**2-Step FID (↓)**
iCT	<u>4.18</u>	2.58
sCM (VE)	5.62	2.73
ECM	4.54	**2.20**
ADCM (Ours)	**3.16**	<u>2.44</u>

D Limitations and Broader Impacts

In this paper, we introduce ADCMs, an adaptive discretization method for CMs. Our approach effectively improves both the training efficiency and generation quality of CMs, and demonstrates adaptability to different variants of DMs. However, ADCMs focus on Consistency Training (CT), as it generally yields better performance. In the case of Consistency Distillation (CD), estimating

$\mathcal{L}_{\text{global}}$ significantly increases training costs due to the need for iterative solving of the endpoint of the PF-ODE. We leave this issue for future work. ADCMs enable efficient content generation for creators while significantly reducing the computational cost of obtaining similar models. Additionally, similar to other deep generative models, ADCMs could be misused to generate harmful fake content, and the proposed method may further exacerbate the potential risks associated with malicious applications of such models.

E License

We list the used datasets, models and their citations, URLs, and licenses in Table 11.

Table 11: Licenses and citations for existing assets.

Name	**URL**	**Citation**	**License**
CIFAR-10	`https://www.cs.toronto.edu/~kriz/cifar.html`	[16]	\
ImageNet	`https://www.image-net.org`	[3]	\
EDM	`https://github.com/NVlabs/edm`	[11]	Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
EDM2	`https://github.com/NVlabs/edm2`	[12]	Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
Rectified Flow	`https://github.com/gnobitab/RectifiedFlow`	[20]	\

F Additional Samples

We provide additional samples of ADCMs from unconditional CIFAR-10 and class-conditional ImageNet 64×64 in Figures 5 - 7.

Figure 5: 1-step samples from unconditional CIFAR-10 trained with a budget of 76.8M images.

Figure 6: 1-step samples from unconditional CIFAR-10 trained with a budget of 12.8M images.

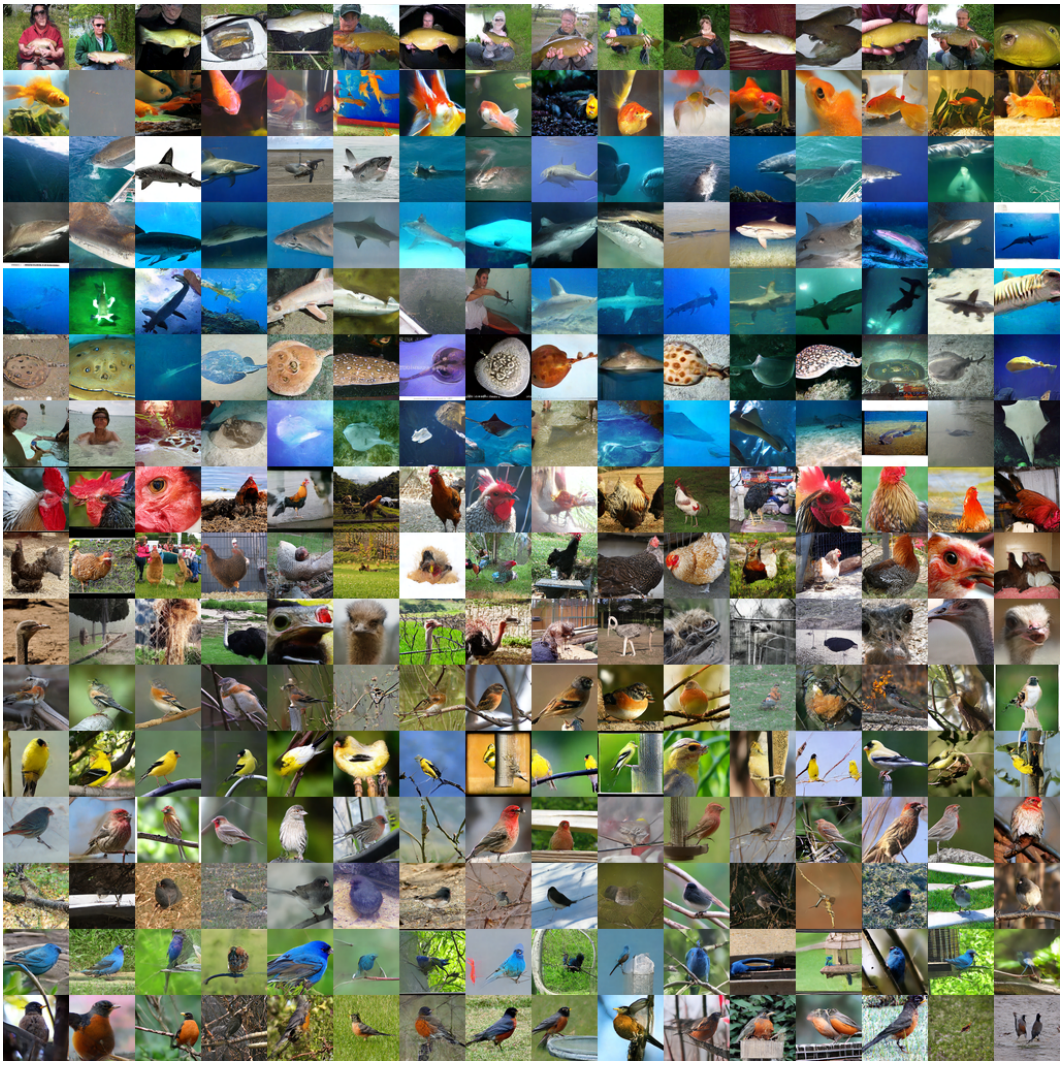


Figure 7: 1-step samples from class-conditional ImageNet 64×64 trained with a budget of 12.8M images.

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: The main claims made in the abstract and introduction accurately reflect the paper's contributions.

   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 discuss the limitations of the work. See sec D.

   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: The paper does not include theoretical results.

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: The paper fully disclose all the information needed to reproduce the main experimental results of the paper. Detailed algorithm can be found in Algorithm 1. Detailed settings can be found in Sec 4 and Appendix A. Code is available in the supplementary material.

   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: The paper provide open access to the code. Code is available in the supplementary material.

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, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The paper specify all the training and test details. Detailed settings can be found in Sec 4 and Appendix A.

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: Since the FID is computed using 50k samples, we find that the standard deviation of ADCMs' FID is very small, and thus it does not affect the conclusions of our experiments.

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: We provide sufficient information on the computer resources in Table 6.

   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: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   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 discuss both potential positive societal impacts and negative societal impacts of the work performed. See Appendix D.

    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: [NA]

    Justification: The paper poses no such risks.

    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: We list the used datasets, models and their citations, URLs and licenses in Appendix E.

    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` 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: [Yes]

    Justification: The code is well documented and the documentation is available in the supplementary material.

    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 core method development in this research does not involve LLMs as any important, original, or non-standard components.

    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.