Phased Consistency Models

Fu-Yun Wang[1] **Zhaoyang Huang**[2] **Alexander William Bergman**[3,6] **Dazhong Shen**[4]
Peng Gao[4] **Michael Lingelbach**[3,6] **Keqiang Sun**[1] **Weikang Bian**[1]
Guanglu Song[5] **Yu Liu**[4] **Xiaogang Wang**[1] **Hongsheng Li**[1,4,7]

[1]MMLab, CUHK [2]Avolution AI [3]Hedra
[4]Shanghai AI Lab [5]Sensetime Research [6]Stanford University [7]CPII under InnoHK

{fywang@link, xgwang@ee, hsli@ee}.cuhk.edu.hk

Abstract

Consistency Models (CMs) have made significant progress in accelerating the generation of diffusion models. However, their application to high-resolution, text-conditioned image generation in the latent space remains unsatisfactory. In this paper, we identify three key flaws in the current design of Latent Consistency Models (LCMs). We investigate the reasons behind these limitations and propose **P**hased **C**onsistency **M**odels (PCMs), which generalize the design space and address the identified limitations. Our evaluations demonstrate that PCMs outperform LCMs across 1–16 step generation settings. While PCMs are specifically designed for multi-step refinement, they achieve comparable 1-step generation results to previously state-of-the-art specifically designed 1-step methods. Furthermore, we show the methodology of PCMs is versatile and applicable to video generation, enabling us to train the state-of-the-art few-step text-to-video generator. Our code is available at `https://github.com/G-U-N/Phased-Consistency-Model`.

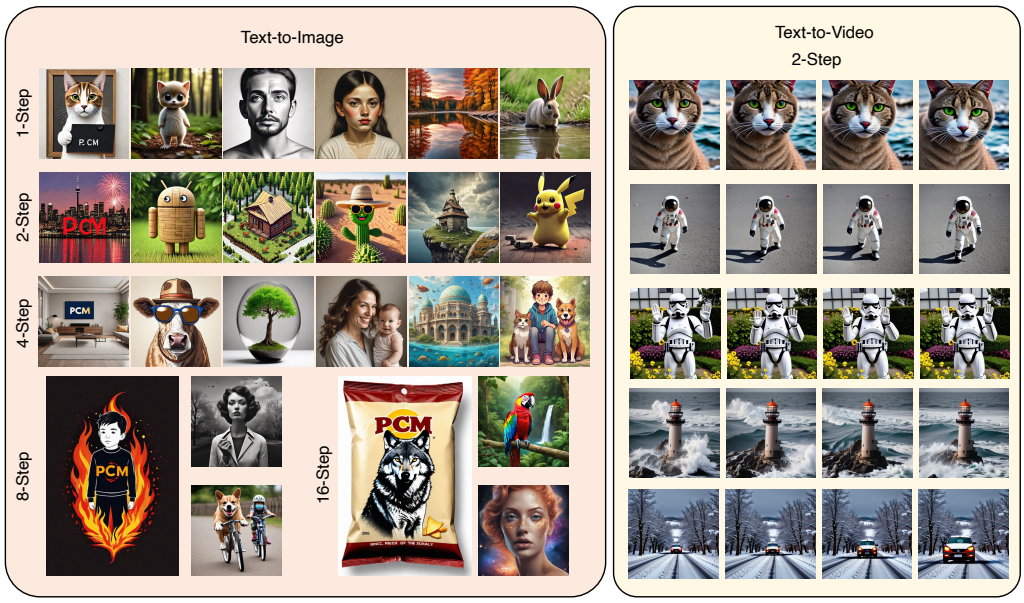

Figure 1: ***PCMs**: Towards stable and fast image and video generation.*

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-2668

Figure 2: **Summative motivation.** We observe and summarize three crucial limitations for (latent) consistency models, and generalize the design space, well tackling all these limitations.

1 Introduction

Diffusion models [20, 15, 57, 68] have emerged as the dominant methodology for image synthesis [45, 41, 7] and video synthesis [22, 53, 52, 63]. These models have shown the ability to generate high-quality and diverse samples conditioned on varying signals. At their core, diffusion models rely on an iterative evaluation to generate new samples. This iterative evaluation trajectory models the probability flow ODE (PF-ODE) [56, 57] that transforms an initial normal distribution to a target real data distribution. However, the iterative nature of diffusion models makes the generation of new samples time-intensive and resource-consuming. To address this challenge, consistency models [56, 11, 63, 55] have emerged to reduce the number of iterative steps required to generate samples. These models work by training a model that enforces the self-consistency [56] property: *any point along the same PF-ODE trajectory shall be mapped to the same solution point.* These models have been extended to high-resolution text-to-image synthesis with latent consistency models (LCMs) [35]. Despite the improvements in efficiency and the ability to generate samples in a few steps, the sample quality of such models is still limited.

We show that the current design of LCMs is flawed, causing inevitable drawbacks in controllability, consistency, and efficiency during image sampling. Fig. 2 illustrates our observations of LCMs. The limitations are listed as follows:

(1) **Consistency.** Due to the specific consistency property, CMs can only use the purely stochastic multi-step sampling algorithm, which assumes that the accumulated noise variable in each generative step is independent and causes varying degrees of stochasticity for different inference-step settings. As a result, we can find inconsistency among the samples generated with the same seeds in different inference steps.

(2) **Controllability.** Even though diffusion models can adopt classifier-free guidance (CFG) [16] in a wide range of values (i.e. 2–15), equipped with weights of LCMs, they can only accept values of CFG within range of 1–2. Larger values of CFG would cause the exposure problem. This brings difficulty for the hyper-parameter selection. Additionally, we find that LCMs are insensitive to the negative prompt. As shown in the figure, LCMs still generate black dogs even when the negative prompt is set to “black dog". Both phenomenon reduce the controllability on generation. We show the reason behind this is the CFG-augmented ODE solver adopted in the consistency distillation stage.

(3) **Efficiency.** LCMs tend to generate much inferior samples at the few-step settings, especially in less than 4 inference steps, which limits the sampling efficiency. We argue that the reason lies in the traditional L2 loss or the Huber loss used in the LCMs procedure, which is insufficient for the fine-grained supervision in few-step settings.

To this end, we propose Phased Consistency Models (PCMs), which can tackle the discussed limitations of LCMs and are easy to train. Specifically, instead of mapping all points along the ODE trajectory to the same solution, PCMs phase the ODE trajectory into several sub-trajectories and only enforce the self-consistency property on each sub-trajectory. Therefore, PCMs can sample

https://doi.org/10.52202/079017-2668

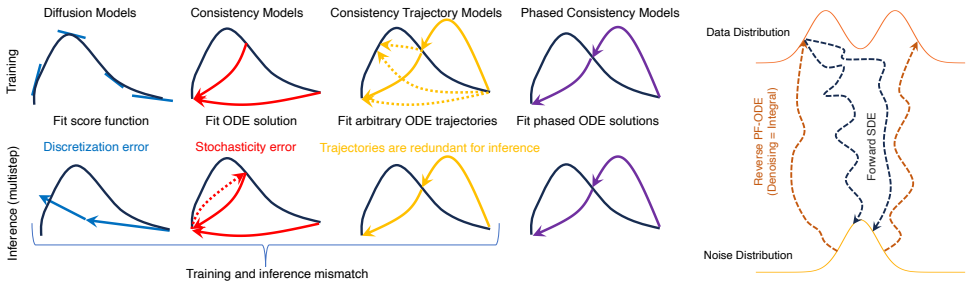

Figure 3: (Left) Illustrative comparison of diffusion models [15], consistency models [56], consistency trajectory models [21], and our phased consistency model. (Right) Simplified visualization of the forward SDE and reverse-time PF-ODE trajectories.

samples along the solution points of different sub-trajectories in a deterministic manner without error accumulation. As an example shown in Fig. 3, we train PCMs with two sub-trajectories, with three edge points including $\mathbf{x}_T$, $\mathbf{x}_0$, and $\mathbf{x}_{\lfloor T/2 \rfloor}$ selected. Thereby, we can achieve 2-step deterministic sampling (i.e., $\mathbf{x}_T \to \mathbf{x}_{\lfloor T/2 \rfloor} \to \mathbf{x}_0$) for generation. Moreover, the phased nature of PCMs leads to an additional advantage. For distillation, we can choose to use a normal ODE solver without the CFG alternatively in the consistency distillation stage, which is not viable in LCMs. As a result, PCMs can optionally use larger values of CFG for inference and be more responsive to the negative prompt. Additionally, to improve the sample quality for few-step settings, we propose an adversarial loss in the latent space for more fine-grained supervision.

To conclude, we dive into the design of (latent) consistency models, analyze the reasons for their unsatisfactory generation characteristics, and propose effective strategies for tackling these limitations. We validate the effectiveness of PCMs on widely recognized image generation benchmarks with stable diffusion v1-5 (0.9 B) [45] and stable diffusion XL (3B) [41] and video generation benchmarks following AnimateLCM [63]. Vast experimental results show the effectiveness of PCMs.

2 Preliminaries

Diffusion models define a forward conditional probability path, with a general representation of $\alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}$ for intermediate distribution $\mathbb{P}_t(\mathbf{x}|\mathbf{x}_0)$ conditioned on $\mathbf{x}_0 \sim \mathbb{P}_0$, which is equivalent to the stochastic differential equation $\mathrm{d}\mathbf{x}_t = f_t \mathbf{x}_t \mathrm{d}t + g_t \mathrm{d}\boldsymbol{w}_t$ with $\boldsymbol{w}_t$ denoting the standard Winer process, $f_t = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}_t}$ and $g_t^2 = \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} - 2\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t^2$. There exists a deterministic flow for reversing the transition, which is represented by the probability flow ODE (PF-ODE) $\mathrm{d}\mathbf{x}_t = \left[f_t \mathbf{x}_t - g_t^2/2\nabla_{\mathbf{x}_t} \log \mathbb{P}_t(\mathbf{x}_t)\right] \mathrm{d}t$. In standard diffusion training, a neural network is typically learned to estimate the score of marginal distribution $\mathbb{P}_t(\mathbf{x}_t)$, which is equivalent to $\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log \mathbb{P}_t(\mathbf{x}) = \mathbb{E}_{\mathbf{x}_0 \sim \mathbb{P}(\mathbf{x}_0|\mathbf{x})} \left[\nabla_{\mathbf{x}} \log \mathbb{P}_t(\mathbf{x}|\mathbf{x}_0)\right]$. Substitue the $\nabla_{\mathbf{x}} \log \mathbb{P}_t(x)$ with $\boldsymbol{s}_{\boldsymbol{\phi}}(x, t)$, and we get the empirical PF-ODE. Famous solvers including DDIM [54], DPM-solver [33], Euler and Heun [20] can be generally perceived as the approximation of the PF-ODE with specific orders and forms of diffusion for sampling in finite discrete inference steps. However, when the number of discrete steps is too small, they face inevitable discretization errors.

Consistency models [56], instead of estimating the score of marginal distributions, learn to directly predict the solution point of ODE trajectory by enforcing the self-consistency property: all points at the same ODE trajectory map to the same solution point. To be specific, given a ODE trajectory $\{\mathbf{x}_t\}_{t\in[\boldsymbol{\epsilon},T]}$, the consistency models $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot, t)$ learns to achieve $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = \mathbf{x}_{\boldsymbol{\epsilon}}$ by enforcing $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t'}, t')$ for all $t, t' \in [\epsilon, T]$. A boundary condition $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_\epsilon, \epsilon) = \mathbf{x}_\epsilon$ is set to guarantee the successful convergence of consistency training. During the sampling procedure, we first sample from the initial distribution $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Then, the final sample can be generated/refined by alternating denoising and noise injection steps, i.e,

$$\hat{\mathbf{x}}_\epsilon^{\tau_k} = \boldsymbol{f}_\theta(\hat{\mathbf{x}}_{\tau_k}, \tau_k), \quad \hat{\mathbf{x}}_{\tau_{k-1}} = \hat{\mathbf{x}}_\epsilon^{\tau_k} + \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{1}$$

where $\{\boldsymbol{\tau}_k\}_{k=1}^K$ are selected time points in the ODE trajectory and $\tau_K = T$. Actually, because only one solution point is used to conduct the consistency model, it is inevitable to introduce stochastic error, i.e., $\boldsymbol{\eta}$, for multiple sampling procedures.

https://doi.org/10.52202/079017-2668

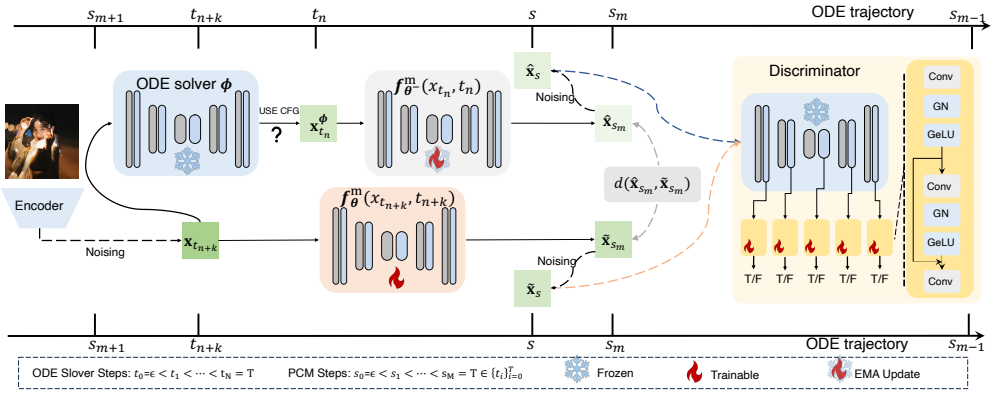

Figure 4: **Training paradigm of PCMs**. '?' means optional usage.

Recently, Consistency trajectory models (CTMs) [21] propose a more flexible framework. specifically, CTMs $\boldsymbol{f_\theta}(\cdot, t, s)$ learns to achieve $\boldsymbol{f_\theta}(\mathbf{x}_t, t, s) = \boldsymbol{f}(x_t, t, s; \boldsymbol{\phi})$ by enforcing $\boldsymbol{f_\theta}(\mathbf{x}_t, t, s) = \boldsymbol{f_\theta}(\mathbf{x}_{t'}, t', s)$ with $s \leq \min(t, t')$ and $t, t' \in [\epsilon, T]$. However, the learning objectives of CTMs are redundant, including many trajectories that will never be applied for inference. More specifically, if we split the continuous time trajectory into N discrete points, diffusion models learn $\mathcal{O}(N)$ objectives (i.e., each point learns to move to its adjacent point), consistency models learn $\mathcal{O}(N)$ objectives (i.e., each point learns to move to solution point), and consistency trajectory models learn $\mathcal{O}(N^2)$ objectives (i.e., each point learns to move to all the other points in the trajectory). Hence, except for the current timestep embedding, CTMs should additionally learn a target timestep embedding, which is not comprised of the design space of diffusion models.

Different from the above approaches, PCMs can be optimized efficiently and support deterministic sampling without additional stochastic error. Overall, Fig. 3 illustrates the difference in training and inference processes among diffusion models, consistency models, consistency trajectory models, and phased consistency models.

3 Method

In this section, we introduce the technical details of PCMs, which overcome the limitations of LCMs in terms of consistency, controllability, and efficiency. **Consistency**: Specifically, we first introduce the main framework of PCMs, consisting of definition, parameterization, the distillation objective, and the sampling procedure in Sec. 3.1. In particular, by enforcing the self-consistency property in multiple ODE sub-trajectories respectively, PCMs can support deterministic sampling to preserve image consistency with varying inference steps. **Controllability**: Secondly, in Sec. 3.2, to improve the controllability of text guidance, we revisit the potential drawback of the guided distillation adopted in LCMs, and propose to optionally remove the CFG for consistency distillation. **Efficiency**: Thirdly, in Sec. 3.3, to further improve inference efficiency, we introduce an adversarial consistency loss to enforce the modeling of data distribution, which facilitates 1-step generation.

3.1 Main Framework

Definition. For a solution trajectory of a diffusion model $\{\mathbf{x}_t\}_{t\in[\epsilon,T]}$ following the PF-ODE, we split the trajectory into multiple sub-trajectories with hyper-defined edge timesteps $s_0, s_1, \ldots, s_M$, where $s_0 = \epsilon$ and $s_M = T$. The M sub-trajectories can be represented as $\{\mathbf{x}_t\}_{t\in[s_m,s_{m+1}]}$ with $m = 0, 1, \ldots, M-1$. We treat each sub-trajectory as an independent CM and define the consistency function as $\boldsymbol{f}^m : (\mathbf{x}_t, t) \rightarrow \mathbf{x}_{s_m}, \quad t \in [s_m, s_{m+1}]$. We learn $\boldsymbol{f}^m_\theta$ to estimate $\boldsymbol{f}$ by enforcing the self-consistency property on each sub-trajectory that its outputs are consistent for arbitrary pairs on the same sub-trajectory. Namely, $\boldsymbol{f}^m(\mathbf{x}_t, t) = \boldsymbol{f}^m(\mathbf{x}_{t'}, t')$ for all $t, t' \in [s_m, s_{m+1}]$. Note that the consistency function is only defined on the sub-trajectory. However, for sampling, it is necessary to define a transition from timestep T (i.e., s_M) to ϵ (i.e., s_0). Thereby, we defined $\boldsymbol{f}^{m,m'} = \boldsymbol{f}^{m'}\left(\cdots \boldsymbol{f}^{m-2}\left(\boldsymbol{f}^{m-1}\left(\boldsymbol{f}^m(\mathbf{x}_t, t), s_m\right), s_{m-1}\right)\cdots, s_{m'}\right)$ that transforms any point $\mathbf{x}_t$ on m-th sub-trajectory to the solution point of m'-th trajectory.

Parameterization. Following the definition, the corresponding consistency function of each sub-trajectory should satisfy boundary condition $\boldsymbol{f}^m(\mathbf{x}_{s_m}, s_m) = \mathbf{x}_{s_m}$, which is crucial for guaranteeing

the successful training of consistency models. To satisfy the boundary condition, we typically need to explicitly parameterize $\boldsymbol{f}_{\boldsymbol{\theta}}^m(\mathbf{x}, t)$ as $\boldsymbol{f}_{\boldsymbol{\theta}}^m(\mathbf{x}_t, t) = c_{\text{skip}}^m(t)\mathbf{x}_r + c_{\text{out}}^m(t)F_{\boldsymbol{\theta}}(\mathbf{x}_t, t, s_m)$, where $c_{\text{skip}}^m(t)$ gradually increases to 1 and $c_{\text{out}}^m(t)$ gradually decays to 0 from timestep s_{m+1} to s_m.

Another important thing is how to parameterize $F_{\boldsymbol{\theta}}(\mathbf{x}_t, t, s)$, basically it should be able to indicate the target prediction at timestep s given the input $\mathbf{x}$ at timestep t. Since we build upon the epsilon prediction models, we hope to maintain the epsilon-prediction learning target.

For the above-discussed PF-ODE, there exists an exact solution [33] from timestep t to s

$$\mathbf{x}_s = \frac{\alpha_s}{\alpha_t}\mathbf{x}_t + \alpha_s \int_{\lambda_t}^{\lambda_s} e^{-\lambda}\sigma_{t_\lambda(\lambda)}\nabla \log \mathbb{P}_{t_\lambda(\lambda)}(\mathbf{x}_{t_\lambda(\lambda)})\mathrm{d}\lambda \tag{2}$$

where $\lambda_t = \ln \frac{\alpha_t}{\sigma_t}$ and t_λ is a inverse function with λ_t. The equation shows that the solution of the ODE from t to s is the scaling of $\mathbf{x}_t$ and the weighted sum of scores. Given a epsilon prediction diffusion network $\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}, t)$, we can estimate the solution as $\mathbf{x}_s = \frac{\alpha_s}{\alpha_t}\mathbf{x}_t - \alpha_s \int_{\lambda_t}^{\lambda_s} e^{-\lambda}\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_{t_\lambda(\lambda)}, t_\lambda(\lambda))\mathrm{d}\lambda$. However, note that the solution requires knowing the epsilon prediction at each timestep between t and s, but consistency modes need to predict the solution with only $\mathbf{x}_t$ available with single network evaluation. Thereby, we parameterize the $F_{\boldsymbol{\theta}}(\mathbf{x}, t, s)$ as following,

$$\mathbf{x}_s = \frac{\alpha_s}{\alpha_t}\mathbf{x}_t - \alpha_s\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t)\int_{\lambda_t}^{\lambda_s} e^{-\lambda}\mathrm{d}\lambda\,. \tag{3}$$

One can show that the parameterization has the same format with DDIM $\mathbf{x}_s = \alpha_s(\frac{\mathbf{x}_t - \sigma_t\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\alpha_t}) + \sigma_s\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ (see Theorem 3). But here we clarify that the parameterization has an intrinsic difference from DDIM. DDIM is the first-order approximation of solution ODE, which works because we assume the linearity of the score in small intervals. This causes the DDIM to degrade dramatically in few-step settings since the linearity is no longer satisfied. Instead, our parameterization is not approximation but exact solution learning. The learning target of $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ is no more the scaled score $-\sigma_t\nabla_{\mathbf{x}}\log\mathbb{P}_t(\mathbf{x})$ (which epsilon-prediction diffusion models learn to estimate) but $\frac{\int_{\lambda_t}^{\lambda_s} e^{-\lambda}\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_{t_\lambda(\lambda)}, t_\lambda(\lambda))\mathrm{d}\lambda}{\int_{\lambda_t}^{\lambda_s} e^{-\lambda}\mathrm{d}\lambda}$. Actually, we can define the parameterization in other formats, but we find this format is simple and has a small gap between the original diffusion models. The parameterization of F_θ also allows for a better property that we can drop the introduced c_{skip}^m and c_{out}^m in consistency models to ease the complexity of the framework. Note that following Eq. 3, we can get $F_{\boldsymbol{\theta}}(\mathbf{x}_{s_m}, s_m, s_m) = \frac{\alpha_{s_m}}{\alpha_{s_m}}\mathbf{x}_{s_m} - 0 = \mathbf{x}_{s_m}$. Therefore, the boundary condition is already satisfied. Hence, we can simply define $\boldsymbol{f}_{\boldsymbol{\theta}}^m(\mathbf{x}, t) = F_{\boldsymbol{\theta}}(\mathbf{x}, t, s_m)$. This parameterization also aligns with several previous diffusion distillation techniques [46, 3, 74] utilizing DDIM format, building a deep connection with previous distillation methods. The difference is that we provide a more fundamental explanation of the meaning and learning objective of parameterizations.

Phased consistency distillation objective. Denote the pre-trained diffusion models as $\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}, t) = -\frac{\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}, t)}{\sigma_t}$, which induces an empirical PF-ODE. We firstly discretize the whole trajectory into N sub-intervals with $N+1$ discrete timesteps from $[\epsilon, T]$, which we denote as $t_0 = \epsilon < t_1 < t_2 < \cdots < t_N = T$. Typically N should be sufficiently large to make sure the ODE solver approximates the ODE trajectory correctly. Then we sample $M+1$ timesteps as edge timesteps $s_0 = t_0 < s_1 < s_2 < \cdots < s_M = t_N \in \{t_i\}_{i=0}^N$ to split the ODE trajectory into M sub-trajectories. Each sub-trajectory $[s_i, s_{i+1}]$ consists of the set of sub-intervals $\{[t_j, t_{j+1}]\}_{t_j \geq s_i, t_{j+1} \leq s_{i+1}}$.

Here we define $\Phi(\mathbf{x}_{t_{n+k}}, t_{n+k}, t_n; \boldsymbol{\phi})$ as the k-step ODE solver that approximate $\mathbf{x}_{t_n}^{\boldsymbol{\phi}}$ from $\mathbf{x}_{t_{n+k}}$ on the same sub-trajectory following Equation 2, namely,

$$\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}} = \Phi(\mathbf{x}_{t_{n+k}}, t_{n+k}, t_n; \boldsymbol{\phi}). \tag{4}$$

Following CMs [56], we set $k = 1$ to minimize the ODE solver cost. The training loss is defined as

$$\mathcal{L}_{\text{PCM}}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) = \mathbb{E}_{\mathbb{P}(m), \mathbb{P}(n|m), \mathbb{P}(\mathbf{x}_{t_{n+1}}|n,m)}\left[\lambda(t_n)d\left(\boldsymbol{f}_{\boldsymbol{\theta}}^m(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}^m(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n)\right)\right] \tag{5}$$

where $\mathbb{P}(m) := \text{uniform}(\{0, 1, \ldots, M-1\})$, $\mathbb{P}(n|m) := \text{uniform}(\{n+1 | t_{n+1} \leq s_{m+1}, t_n \geq s_m\})$, $\mathbb{P}(\mathbf{x}_{t_{n+1}}|n, m) = \mathbb{P}_{t_{n+1}}(\mathbf{x})$, and $\boldsymbol{\theta}^- = \mu\boldsymbol{\theta}^- + (1-\mu)\boldsymbol{\theta}$.

https://doi.org/10.52202/079017-2668

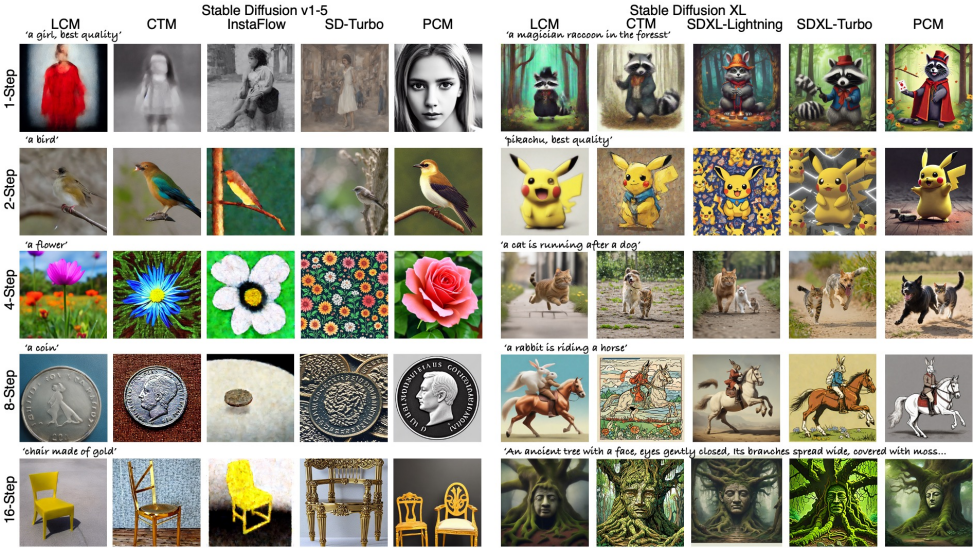

Figure 5: **Qualitative Comparison.** Our method achieves top-tier performance.

We show that when $\mathcal{L}_{\text{PCM}}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) = 0$ and local errors of ODE solvers uniformly bounded by $\mathcal{O}((\Delta t)^{p+1})$, the solution estimation error within the arbitrary sub-trajectory $\boldsymbol{f}_{\boldsymbol{\theta}}^m(\mathbf{x}_{t_n}, t_n)$ is bounded by $\mathcal{O}((\Delta t)^p)$ in Theorem 1. Additionally, the solution estimation error across any sets of sub-trajectories (i.e., the error between $\boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}(\mathbf{x}_{t_n}, t_n)$ and $\boldsymbol{f}^{m,m'}(\mathbf{x}_{t_{n+1},t_{n+1}}; \boldsymbol{\phi})$) is also bounded by $\mathcal{O}((\Delta t)^p)$ in Theorem 2.

Sampling. For a given initial sample at timestep t which belongs to the sub-trajectory $[s_m, s_{m+1}]$, we can support deterministic sampling following the definition of $\boldsymbol{f}^{m,0}$. Previous work [21, 67] reveals that introducing a certain degree of stochasticity might lead to better generation quality. We show that our sampling method can also introduce randomness through a simple modification, which we discuss at Sec. IV.1

3.2 Guided Distillation

For convenience, we take the epsilon-prediction format with text conditions $\boldsymbol{c}$ for the following discussion. The consistency model is denoted as $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, \boldsymbol{c})$, and the diffusion model is denoted as $\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \boldsymbol{c})$. A commonly applied strategy for text-conditioned diffusion models is classifier-free guidance (CFG) [16, 51, 1]. At training, $\boldsymbol{c}$ is randomly substituted with null text embedding $\varnothing$. At each inference step, the model computes $\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \boldsymbol{c})$ and $\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \boldsymbol{c}_{\text{neg}})$ simultaneously, and the actual prediction is the linear combination of them. Namely,

$$\boldsymbol{\epsilon}_{\phi}(\mathbf{x}_t, t, \boldsymbol{c}, \boldsymbol{c}_{\text{neg}}; w) = \boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \boldsymbol{c}_{\text{neg}}) + w(\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \boldsymbol{c}) - \boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \boldsymbol{c}_{\text{neg}})), \tag{6}$$

where w controlling the strength and $\boldsymbol{c}_{\text{neg}}$ can be set to $\varnothing$ or text embeddings of unwanted characteristics. A noticeable phenomenon is that diffusion models with this strategy can not generate content with good quality without using CFG. That is, the empirical ODE trajectory induced with pure $\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \boldsymbol{c})$ deviates away from the ODE trajectory to real data distribution. Thereby, it is necessary to apply CFG-augmented prediction $\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \boldsymbol{c}, \boldsymbol{c}_{\text{neg}}; w)$ for ODE solvers. Recall that we have shown that the consistency learning target of the consistency model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, \boldsymbol{c})$ is the weighted sum of epsilon prediction on the trajectory. Thereby, we have $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, \boldsymbol{c}) \propto \boldsymbol{\epsilon}_{\phi}(\mathbf{x}_{t'}, t', \boldsymbol{c}, \varnothing; w)$, for all $t' \leq t$. On this basis, if we additionally apply the CFG for the consistency models, we can prove that

$$\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, \boldsymbol{c}, \boldsymbol{c}_{\text{neg}}; w') \propto ww'(\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_{t'}, t', \boldsymbol{c}) - \boldsymbol{\epsilon}_{\boldsymbol{\phi}}^{\text{merge}})) + \boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_{t'}, t', \boldsymbol{c}_{\text{neg}}), \tag{7}$$

where $\boldsymbol{\epsilon}_{\boldsymbol{\phi}}^{\text{merge}} = (1-\alpha)\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_{t'}, t', \boldsymbol{c}_{\text{neg}}) + \alpha\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_{t'}, t', \varnothing)$ and $\alpha = \frac{(w-1)}{ww'}$ (See Theorem 4). This equation indicates that applying CFG w' to the consistency models trained with CFG-augmented ODE solver confined with w, is equivalent to scaling the prediction of original diffusion models by $w'w$, which explains the the exposure problem. We can also observe that the epsilon prediction with negative prompts is *diluted* by the prediction with null text embedding, which reveals that the impact of negative prompts is reduced.

Table 1: Comparison of FID-SD and FID-CLIP with Stable Diffusion v1-5 based methods under different steps.

METHODS	FID-SD										FID-CLIP				
	COCO-30K					CC12M-30K					COCO-30K				
	1	2	4	8	16	1	2	4	8	16	1	2	4	8	16
InstaFlow [31]	11.51	69.79	102.15	122.20	139.29	11.90	62.48	88.64	105.34	113.56	**9.56**	29.38	38.24	43.60	47.32
SD-Turbo [49]	10.62	12.22	16.66	24.30	30.32	10.35	12.03	15.15	19.91	23.34	12.08	15.07	15.12	14.90	15.09
LCM [35]	53.43	11.03	6.66	6.62	7.56	42.67	10.51	6.40	5.99	9.02	21.04	10.18	10.64	12.15	13.85
CTM [21]	63.55	9.93	9.30	15.62	21.75	28.47	8.98	8.22	13.27	17.43	30.39	10.32	10.31	11.27	12.75
Ours	**8.27**	**9.79**	**5.81**	**5.00**	**4.70**	**7.91**	**8.93**	**4.93**	**3.88**	**3.85**	11.66	9.17	9.07	9.49	10.13
Ours*	-	-	7.46	6.49	5.78	-	-	4.99	5.01	5.13	-	-	**8.85**	**8.33**	**8.09**

We ask the question: *Is it possible to conduct consistency distillation with diffusion models trained for CFG usage without applying CFG-augmented ODE solver?* Our finding is that it is not applicable for original CMs but works well for PCMs especially when the number of sub-trajectory M is large. We empirically find that $M = 4$ is sufficient for successful training. As we discussed, the text-conditioned diffusion models trained for CFG usage fail at achieving good generation quality when removing CFG for inference. That is, target data distribution induced by PF-ODE with $\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \boldsymbol{c})$ has a large distribution distance to real data distribution. Therefore, fitting the spoiled ODE trajectory is only to make the generation quality bad. In contrast, when phasing the whole ODE trajectory into several sub-trajectories, the negative influence is greatly alleviated. On one hand, the starting point $\mathbf{x}_{s_{m+1}}$ of sub-trajectories is replaced by adding noise to the real data. On the other hand, the distribution distance between the distribution of solution points $\mathbf{x}_{s_m}$ of sub-trajectories and real data distribution $\mathbb{P}^{\text{data}}_{s_m}$ at the same timestep is much smaller proportional to the noise level introduced. To put it straightforwardly, even though the distribution gap between the real data and the samples generated from $\boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \boldsymbol{c})$ is large, adding noise to them reduce the gap. Thereby, we can optionally train a consistency model whether supporting larger values of CFG or not.

3.3 Adversarial Consistency Loss

We introduce an adversarial loss to enforce the distribution consistency, which greatly improves the generation quality in few-step settings. For convenience, we introduce an additional symbol $\mathcal{T}_{t\to s}$ which represents a flow from $\mathbb{P}_t$ to $\mathbb{P}_s$. Let $\mathcal{T}^{\boldsymbol{\phi}}_{t\to s}$, $\mathcal{T}^{\boldsymbol{\theta}}_{t\to s}$ be the transition mapping following the ODE trajectory of pre-trained diffusion and our consistency models. Additionally, let $\mathcal{T}^{-}_{s\to t}$ be the distribution transition following the forward process SDE (adding noise). The loss function is defined as the following

$$\mathcal{L}^{adv}_{\text{PCM}}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}, m) = D\left(\mathcal{T}^{-}_{s_m\to s}\mathcal{T}^{\boldsymbol{\theta}}_{t_{n+k}\to s_m}\#\mathbb{P}_{t_{n+k}} \middle\| \mathcal{T}^{-}_{s_m\to s}\mathcal{T}^{\boldsymbol{\theta}^-}_{t_n\to s_m}\mathcal{T}^{\boldsymbol{\phi}}_{t_{n+1}\to t_n}\#\mathbb{P}_{t_{n+1}}\right), \quad (8)$$

where $\#$ is the pushforward operator, and D is the distribution distance metric. To penalize the distribution distance, we apply the GAN-style training paradigm. To be specific, as shown in Fig. 4, for the sampled $\mathbf{x}_{t_{n+k}}$ and the $\mathbf{x}^{\boldsymbol{\phi}}_{t_n}$ solved through the pre-trained diffusion model $\boldsymbol{\phi}$, we first compute their predicted solution point $\tilde{\mathbf{x}}_{s_m} = \boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+k}}, t_{n+k})$ and $\hat{\mathbf{x}}_{s_m} = \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}^{\boldsymbol{\phi}}_{t_n}, t_n)$. Then we randomly add noise to $\tilde{\mathbf{x}}_{s_m}$ and $\hat{\mathbf{x}}_{s_m}$ to obtain $\tilde{\mathbf{x}}_s$ and $\hat{\mathbf{x}}_s$ with randomly sampled $s \in [s_m, s_{m+1}]$. We optimize the adversarial loss between $\tilde{\mathbf{x}}_s$ and $\hat{\mathbf{x}}_s$. Specifically, the $\mathcal{L}^{adv}_{\text{PCM}}$ can be re-written as

$$\mathcal{L}^{adv}_{\text{PCM}}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}, m) = \text{ReLU}(1 + \mathcal{D}(\tilde{\mathbf{x}}_s, s, \boldsymbol{c})) + \text{ReLU}(1 - \mathcal{D}(\hat{\mathbf{x}}_s, s, \boldsymbol{c})), \quad (9)$$

where $\text{ReLU}(x) = x$ if $x > 0$ else $\text{ReLU}(x) = 0$, $\mathcal{D}$ is the discriminator, $\boldsymbol{c}$ is the image conditions (e.g., text prompts), and the loss is updated in a min-max manner [12]. Therefore the eventual optimization objective is $\mathcal{L}_{\text{PCM}} + \lambda\mathcal{L}^{adv}_{\text{PCM}}$ with λ as a hyper-parameter controlling the trade-off of distribution consistency and instance consistency. We adopt $\lambda = 0.1$ for all training settings. However, we hope to clarify that the adversarial loss has an intrinsic difference from the GAN. The GAN training aims to align the training data distribution and the generation distribution of the model. That is, $D(\mathcal{T}^{\boldsymbol{\theta}}_{t_{n+k}\to s_m}\#\mathbb{P}_{t_{n+k}} \| \mathbb{P}_{s_m})$ In Theorem 5, we show that consistency property is enforced, our introduced adversarial loss will also coverage to zero. Yet in Theorem 6, we show that combining standard GAN with consistency distillation is a flawed design when considering the pre-trained data distribution and distillation data distribution mismatch. Its loss will be non-zero when the self-consistency property is achieved, thus corrupting the consistency distillation learning. Our experimental results also verify our statement (Fig. 7).

https://doi.org/10.52202/079017-2668

Table 2: One-step generation comparison on Stable Diffusion v1-5.

METHODS	COCO-30K				CC12M-30K				CONSISTENCY ↑
	FID ↓	FID-CLIP ↓	FID-SD ↓	CLIP SCORE ↑	FID ↓	FID-CLIP ↓	FID-SD ↓	CLIP SCORE ↑	
InstaFlow [31]	**13.59**	**9.56**	11.51	<u>29.37</u>	<u>15.06</u>	<u>6.16</u>	<u>11.90</u>	24.52	0.61
SD-Turbo [49]	<u>16.56</u>	12.08	<u>10.62</u>	**31.21**	17.17	6.18	12.48	<u>26.30</u>	0.71
CTM [21]	67.55	30.39	63.55	23.98	56.39	28.47	56.34	18.81	0.65
LCM [35]	53.81	21.04	53.43	25.23	44.35	18.58	42.67	20.38	0.62
TCD [74]	71.69	31.69	68.04	23.60	57.97	30.03	57.21	18.57	-
Ours	17.91	<u>11.66</u>	**8.27**	29.26	**14.79**	**5.38**	**7.91**	**26.33**	**0.81**

4 Experiments

4.1 Experimental Setup

Dataset. Training dataset: For image generation, we train all models on the CC3M [5] dataset. For video generation, we train the model on WebVid-2M [2]. **Evaluation dataset:** For image generation, we evaluate the performance on the COCO-2014 [28] following the 30K split of karpathy. We also evaluate the performance on the CC12M with our randomly chosen 30K split. For video generation, we evaluate with the captions of UCF-101 [58].

Backbones. We verify the text-to-image generation based on Stable Diffusion v1-5 [45] and Stable Diffusion XL [41]. We verify the text-to-video generation following the design of AnimateLCM [63] with decoupled consistency distillation.

Evaluation metrics. Image: We report the FID [14] and CLIP score [43] of the generated images and the validation 30K-sample splits. Following [8, 47], we also compute the FID with CLIP features (FID-CLIP). Note that, all baselines and our method focus on distilling the knowledge from the pre-trained diffusion models for acceleration. Therefore, we also compute the FID of all baselines and the generated images of original pre-trained diffusion models including Stable Diffusion v1-5 and Stable Diffusion XL (FID-SD). **Video:** For video generation, we evaluate the performance from three perspectives: the CLIP Score to measure the text-video alignment, the CLIP Consistency to measure the inter-frame consistency of the generated videos, the Flow Magnitude to measure the motion magnitude of the generated videos with Raft-Large [59].

4.2 Comparison

Comparison methods. We compare PCM with Stable Diffusion v1-5 to methods including Stable Diffusion v1-5 [45], InstaFlow [31], LCM [35], CTM [21], TCD [74] and SD-Turbo [49]. We compare PCM with Stable Diffusion XL to methods including Stable Diffusion XL [41], CTM [21], SDXL-Lightning [27], SDXL-Turbo [49], and LCM [35]. We apply the 'Ours' and 'Ours*' to denote our methods trained with CFG-augmented ODE solver or not. We only report the performance of 'Ours*' with more than 4 steps which aligns with our claim that it is only possible when phasing the ODE trajectory into multiple sub-trajectories. For video generation, we compare with DDIM [54], DPM [33], and AnimateLCM [63].

Qualitative comparison. We evaluate our model and comparison methods with a diverse set of prompts in different inference steps. The results are listed in Fig. 5. Our method shows clearly the top performance in both image visual quality and text-image alignment across 1–16 steps.

Quantitative comparison. One-step generation: We show the one-step generation results comparison of methods based on Stable Diffusion v1-5 and Stable Diffusion XL in Table 2 and Table 5, respectively. Notably, PCM consistently surpasses the consistency model-based methods including LCM and CTM by a large margin. Additionally, it achieves comparable or even superior to the state-of-the-art GAN-based (SD-Turbo, SDXL-Turbo, SDXL-Lightning) or Rectified-Flow-based (InstaFlow) one-step generation methods. Note that InstaFlow applies the LIPIPS [72] loss for training and SDXL-Turbo can only generate 512×512 resolution images, therefore it is easy for them to obtain higher scores. **Multi-step generation:** We report the FID changes of different methods on COCO-30K and CC12M-30K in Table 1 and Table 3. 'Ours' and 'Ours*' achieve the best or second-best performance in most cases. It is notably the gap of performance between our methods and other baselines becoming large as the timestep increases, which indicates the phased nature of our methods supports more powerful multi-step sampling ability. **Video generation:** We show the quantitative comparison of video generation in Table 4, our model achieves consistent superior

Table 3: Comparison of FID-SD on CC12M-30K with Stable Diffusion XL.

METHODS	1-Step	2-Step	4-Step	8-Step	16-Step
SDXL-Lightning [27]	8.69	7.29	**5.26**	5.83	6.10
SDXL-Turbo (512 × 512) [49]	**6.64**	**6.53**	7.39	13.88	26.55
SDXL-LCM [35]	57.70	19.64	11.22	12.89	23.33
SDXL-CTM [21]	72.45	24.06	20.67	39.89	39.18
Ours	8.76	7.02	6.59	**5.19**	**4.92**
Ours*	-	-	6.26	5.27	5.09

Table 4: Quantitative comparison for video generation.

Methods	CLIP Score ↑			Flow Magnitude ↑			CLIP Consistency ↑		
	1-Step	2-Step	4-Step	1-Step	2-Step	4-Step	1-Step	2-Step	4-Step
DDIM [54]	4.44	7.09	23.05	-	-	1.47	-	-	0.877
DPM [33]	11.21	17.93	28.57	-	-	1.95	-	-	0.947
AnimateLCM [63]	25.41	29.39	30.62	1.10	1.81	2.40	**0.967**	0.957	0.965
Ours	**29.88**	**30.22**	**30.72**	**4.56**	**4.38**	**4.69**	0.956	**0.962**	**0.968**

Table 5: One-step and two-step generation comparison on Stable Diffusion XL.

METHODS	COCO-30K (one-step)				CC12M-30K (two-step)				CONSISTENCY ↑
	FID ↓	FID-CLIP ↓	FID-SD ↓	CLIP SCORE ↑	FID ↓	FID-CLIP ↓	FID-SD ↓	CLIP SCORE ↑	
SDXL-Turbo (512 × 512) [49]	19.84	13.56	9.40	**32.31**	**15.36**	**5.26**	**6.53**	**27.91**	0.74
SDXL-Lightning [27]	**19.73**	13.33	**9.11**	30.81	17.99	7.39	7.29	26.31	0.76
SDXL-LCM [35]	74.65	31.63	74.46	27.29	25.88	10.36	19.64	25.84	0.66
SDXL-CTM [21]	82.14	37.43	88.20	26.48	32.05	12.50	24.06	24.79	0.66
Ours	21.23	13.66	9.32	31.55	17.87	5.67	7.02	27.10	**0.83**

performance. The 1-step and 2-step generation results of DDIM and DPM are very noisy, therefore it is meaningless to evaluate their Flow Magnitude and CLIP Consistency.

Human evaluation metrics. To more comprehensively reflect the performance of phased consistency models, we conduct a thorough evaluation using human aesthetic preference metrics, encompassing 1–16 steps. This assessment employs well-regarded metrics, including HPSv2 (HPS) [66], PickScore (PICKSCORE) [23], and Laion Aesthetic Score (AES) [50], to benchmark our method against all comparative baselines. As shown in Table 6 and Table 7, across all evaluated settings, our method consistently achieves either superior or comparable results, with a marked performance advantage over the consistency model baseline LCM, demonstrating its robustness and appeal across diverse human-centric evaluation criteria. We conduct a human preference ablation study on the proposed adversarial consistency loss, with the results presented in Table 8. The inclusion of adversarial consistency loss consistently enhances human evaluation metrics across different inference steps.

Table 6: Aesthetic evaluation on SD v1-5.

Steps	Methods	HPS	AES	PICKSCORE
1	InstaFlow	0.267	5.010	0.207
	SD-Turbo	0.276 (1)	5.445 (1)	0.223 (1)
	CTM	0.240	5.155	0.195
	LCM	0.251	5.178	0.201
	Ours	0.276 (1)	5.389 (2)	0.213 (2)
2	InstaFlow	0.249	5.050	0.196
	SD-Turbo	0.278 (1)	5.570 (1)	0.226 (1)
	CTM	0.267	5.117	0.208
	LCM	0.266	5.135	0.210
	Ours	0.275 (2)	5.370 (2)	0.217 (2)
4	InstaFlow	0.243	4.765	0.192
	SD-Turbo	0.278 (2)	5.537 (1)	0.224 (1)
	CTM	0.274	5.189	0.213
	LCM	0.273	5.264	0.215
	Ours	0.279 (1)	5.412 (2)	0.217 (2)
8	InstaFlow	0.267	4.548	0.189
	SD-Turbo	0.276 (2)	5.390 (2)	0.221 (1)
	CTM	0.271	5.026	0.210
	LCM	0.274	5.366	0.216
	Ours	0.278 (1)	5.398 (1)	0.218 (2)
16	InstaFlow	0.237	4.437	0.187
	SD-Turbo	0.277 (1)	5.275	0.219 (1)
	CTM	0.270	4.870	0.209
	LCM	0.274	5.352 (2)	0.216
	Ours	0.277 (1)	5.442 (1)	0.217 (2)

Table 7: Aesthetic evaluation on SDXL.

Steps	Methods	HPS	AES	PICKSCORE
1	SDXL-Lightning	0.278	5.65 (1)	0.223
	SDXL-Turbo	0.279 (1)	5.40	0.228 (1)
	SDXL-CTM	0.239	4.86	0.201
	SDXL-LCM	0.205	5.04	0.206
	Ours	0.280 (1)	5.62 (2)	0.225 (2)
2	SDXL-Lightning	0.280	5.72 (1)	0.227 (1)
	SDXL-Turbo	0.281 (2)	5.46	0.226 (2)
	SDXL-CTM	0.267	5.58	0.216
	SDXL-LCM	0.265	5.40	0.217
	Ours	0.282 (1)	5.688 (2)	0.225
4	SDXL-Lightning	0.281	5.76 (2)	0.228 (1)
	SDXL-Turbo	0.284 (1)	5.49	0.224
	SDXL-CTM	0.278	5.84 (1)	0.221
	SDXL-LCM	0.274	5.48	0.223
	Ours	0.284 (1)	5.645	0.228 (2)
8	SDXL-Lightning	0.282	5.75 (2)	0.229 (1)
	SDXL-Turbo	0.283 (2)	5.59	0.225
	SDXL-CTM	0.276	5.88 (1)	0.218
	SDXL-LCM	0.277	5.57	0.223
	Ours	0.285 (1)	5.676	0.229 (2)
16	SDXL-Lightning	0.280 (2)	5.72 (2)	0.225 (2)
	SDXL-Turbo	0.277	5.56	0.219
	SDXL-CTM	0.274	5.85 (1)	0.215
	SDXL-LCM	0.276	5.64	0.221
	Ours	0.284 (1)	5.646	0.228 (1)

Table 8: Aesthetic ablation study on the adversarial consistency loss.

Methods	Step 1			Step 2			Step 4			Step 8			Step 16		
	HPS	AES	PICKSCORE	HPS	AES	PICKSCORE	HPS	AES	PICKSCORE	HPS	AES	PICKSCORE	HPS	AES	PICKSCORE
PCM w/ adv	0.280	5.620	0.225	0.282	5.688	0.225	0.284	5.645	0.228	0.285	5.676	0.229	0.284	5.646	0.228
PCM w/o adv	0.251	4.994	0.206	0.275	5.502	0.220	0.281	5.576	0.225	0.283	5.637	0.227	0.283	5.620	0.227

4.3 Ablation Study

Sensitivity to negative prompt. To show the comparison of sensitivity to negative prompt between our model tuned without CFG-augmented ODE solver and LCM. We provide an example of a prompt

https://doi.org/10.52202/079017-2668

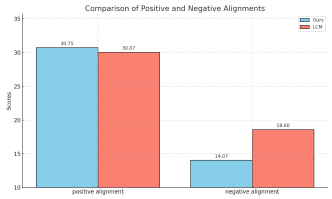

Figure 6: Sensitivity to negative prompt.

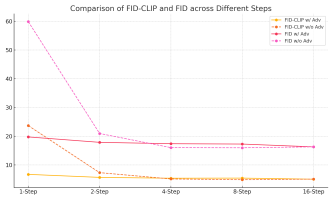

Figure 7: Effectiveness of adversarial consistency loss.

Figure 8: Randomness for sampling.

Figure 9: **Ablation study on the adversarial consistency design.** (Left) Replacing latent discriminator with DINO causes detail loss. (Right) Replacing our adversarial loss with normal GAN loss causes training conflicted objectives and instability.

and negative prompt to GPT-4o and ask it to generate 100 pairs of prompts and their corresponding negative prompts. For each prompt, we generate 10 images. We first generate images without using the negative prompt to show the positive prompt alignment comparison. Then we generate images with positive and negative prompts. We compute the CLIP score of generated images and the prompts and negative prompts. Fig. 6 shows that we not only achieve better prompt alignment but are much more sensitive to negative prompts.

Consistent generation ability. Consistent generation ability under different inference steps is valuable in practice for multistep refinement. We compute the average CLIP similarity between the 1-step generation and the 16-step generation for each method. As shown in the rightmost column of Table 2 and Table 5, our method achieves significantly better consistent generation ability.

Adversarial consistency design and its effectiveness. We show the ablation study on the adversarial consistency loss design and its effectiveness. **From the architecture level of discriminator**, we compare the latent discriminator shared from the teacher diffusion model and the pixel discriminator from pre-trained DINO [4]. Note that DINO is trained with 224 resolutions, therefore we should resize the generation results and feed them into DINO. We find this could make the generation results fail at details as shown in the left of Fig. 9. **From the adversarial loss**, we compare our adversarial loss to the normal GAN loss. We find normal GAN loss causes the training to be unstable and corrupts the generation results, which aligns with our previous analysis. **For its effectiveness**, we compare the FID-CLIP and FID scores with the adversarial loss or without the adversarial loss under different inference steps. Fig. 7 shows that it greatly improves the FID scores in the low-step regime and gradually coverage to similar performance of our model without using the adversarial loss as the step increases.

Randomness for sampling. Fig. 8 illustrates the influence of the randomness introduced in sampling as Eq. 44. The figure shows that introducing a certain of randomness in sampling may help to alleviate unrealistic objects or shapes.

5 Limitations and Conclusions

Despite being able to generate high-quality images and videos in a few steps, we find that when the number of steps is very low, especially with only one step, the generation quality is unstable. The model may produce structural errors or blurry images. Fortunately, we discover that this phenomenon can be mitigated through multi-step refinement. In conclusion, in this paper, we observe the defects in latent consistency models. We summarize these defects on three levels, analyze their causes, and generalize the design framework to address these defects.

https://doi.org/10.52202/079017-2668

6 Acknowledgements

This project is funded in part by National Key R&D Program of China Project 2022ZD0161100, by the Centre for Perceptual and Interactive Intelligence (CPII) Ltd under the Innovation and Technology Commission (ITC)'s InnoHK, by General Research Fund of Hong Kong RGC Project 14204021. Hongsheng Li is a PI of CPII under the InnoHK.

References

[1] Donghoon Ahn, Hyoungwon Cho, Jaewon Min, Wooseok Jang, Jungwoo Kim, SeonHwa Kim, Hyun Hee Park, Kyong Hwan Jin, and Seungryong Kim. Self-rectifying diffusion sampling with perturbed-attention guidance. *arXiv preprint arXiv:2403.17377*, 2024.

[2] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *International Conference on Computer Vision*, pages 1728–1738, 2021.

[3] 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.

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *International Conference on Computer Vision*, 2021.

[5] Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *Conference on Computer Vision and Pattern Recognition*, pages 3558–3568, 2021.

[6] Ricky TQ Chen and Yaron Lipman. Riemannian flow matching on general geometries. *arXiv preprint arXiv:2302.03660*, 2023.

[7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[8] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024.

[9] Peng Gao, Le Zhuo, Ziyi Lin, Chris Liu, Junsong Chen, Ruoyi Du, Enze Xie, Xu Luo, Longtian Qiu, Yuhang Zhang, et al. Lumina-t2x: Transforming text into any modality, resolution, and duration via flow-based large diffusion transformers. *arXiv preprint arXiv:2405.05945*, 2024.

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

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

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

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

[14] 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.

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. ddpm. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

 https://doi.org/10.52202/079017-2668

[16] Jonathan Ho and Tim Salimans. cfg. *arXiv preprint arXiv:2207.12598*, 2022.

[17] Minghui Hu, Jianbin Zheng, Chuanxia Zheng, Chaoyue Wang, Dacheng Tao, and Tat-Jen Cham. One more step: A versatile plug-and-play module for rectifying diffusion schedule flaws and enhancing low-frequency controls. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7331–7340, June 2024.

[18] Dongzhi Jiang, Guanglu Song, Xiaoshi Wu, Renrui Zhang, Dazhong Shen, Zhuofan Zong, Yu Liu, and Hongsheng Li. Comat: Aligning text-to-image diffusion model with image-to-text concept matching. *arXiv preprint arXiv:2404.03653*, 2024.

[19] Minguk Kang, Richard Zhang, Connelly Barnes, Sylvain Paris, Suha Kwak, Jaesik Park, Eli Shechtman, Jun-Yan Zhu, and Taesung Park. Distilling diffusion models into conditional gans. *arXiv preprint arXiv:2405.05967*, 2024.

[20] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. edm. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.

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

[22] Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. On density estimation with diffusion models. In *Advances in Neural Information Processing Systems*, 2021.

[23] Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:36652–36663, 2023.

[24] Fei Kong, Jinhao Duan, Lichao Sun, Hao Cheng, Renjing Xu, Hengtao Shen, Xiaofeng Zhu, Xiaoshuang Shi, and Kaidi Xu. Act-diffusion: Efficient adversarial consistency training for one-step diffusion models, 2024.

[25] Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. Cllms: Consistency large language models, 2024.

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

[27] Shanchuan Lin, Anran Wang, and Xiao Yang. Sdxl-lightning: Progressive adversarial diffusion distillation. *arXiv preprint arXiv:2402.13929*, 2024.

[28] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *European Conference on Computer Vision*, volume 8693, pages 740–755. Springer, 2014.

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

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

[31] Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, et al. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *International Conference on Learning Representations*, 2023.

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

[33] 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.

[34] Guanxing Lu, Zifeng Gao, Tianxing Chen, Wenxun Dai, Ziwei Wang, and Yansong Tang. Manicm: Real-time 3d diffusion policy via consistency model for robotic manipulation. *arXiv preprint arXiv:2406.01586*, 2024.

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

[36] Bingqi Ma, Zhuofan Zong, Guanglu Song, Hongsheng Li, and Yu Liu. Exploring the role of large language models in prompt encoding for diffusion models, 2024.

[37] Xiaofeng Mao, Zhengkai Jiang, Fu-Yun Wang, Wenbing Zhu, Jiangning Zhang, Hao Chen, Mingmin Chi, and Yabiao Wang. Osv: One step is enough for high-quality image to video generation. *arXiv preprint arXiv:2409.11367*, 2024.

[38] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Conference on Computer Vision and Pattern Recognition*, pages 14297–14306, 2023.

[39] Zachary Novack, Ge Zhu, Jonah Casebeer, Julian McAuley, Taylor Berg-Kirkpatrick, and Nicholas J Bryan. Presto! distilling steps and layers for accelerating music generation. *arXiv preprint arXiv:2410.05167*, 2024.

[40] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, October 2023.

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

[42] Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation, 2024.

[43] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Learning Representations*, pages 8748–8763. PMLR, 2021.

[44] Yuxi Ren, Xin Xia, Yanzuo Lu, Jiacheng Zhang, Jie Wu, Pan Xie, Xing Wang, and Xuefeng Xiao. Hyper-sd: Trajectory segmented consistency model for efficient image synthesis. *arXiv preprint arXiv:2404.13686*, 2024.

[45] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

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

[47] Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster. *Advances in Neural Information Processing Systems*, 34:17480–17492, 2021.

[48] Axel Sauer, Tero Karras, Samuli Laine, Andreas Geiger, and Timo Aila. Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis. In *International Conference on Learning Representations*, pages 30105–30118. PMLR, 2023.

[49] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023.

[50] Christoph Schuhmann. Laion-aesthetics. `https://laion.ai/blog/laion-aesthetics/`, 2022. Accessed: 2023-11-10.

 https://doi.org/10.52202/079017-2668

[51] Dazhong Shen, Guanglu Song, Zeyue Xue, Fu-Yun Wang, and Yu Liu. Rethinking the spatial inconsistency in classifier-free diffusion guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9370–9379, 2024.

[52] Xiaoyu Shi, Zhaoyang Huang, Fu-Yun Wang, Weikang Bian, Dasong Li, Yi Zhang, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, et al. Motion-i2v: Consistent and controllable image-to-video generation with explicit motion modeling. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.

[53] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

[54] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

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

[56] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

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

[58] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.

[59] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow, 2020.

[60] Cunzheng Wang, Ziyuan Guo, Yuxuan Duan, Huaxia Li, Nemo Chen, Xu Tang, and Yao Hu. Target-driven distillation: Consistency distillation with target timestep selection and decoupled guidance. *arXiv preprint arXiv:2409.01347*, 2024.

[61] Fu-Yun Wang, Wenshuo Chen, Guanglu Song, Han-Jia Ye, Yu Liu, and Hongsheng Li. Gen-l-video: Multi-text to long video generation via temporal co-denoising. *arXiv preprint arXiv:2305.18264*, 2023.

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

[63] Fu-Yun Wang, Zhaoyang Huang, Xiaoyu Shi, Weikang Bian, Guanglu Song, Yu Liu, and Hongsheng Li. Animatelcm: Accelerating the animation of personalized diffusion models and adapters with decoupled consistency learning. *arXiv preprint arXiv:2402.00769*, 2024.

[64] Fu-Yun Wang, Xiaoshi Wu, Zhaoyang Huang, Xiaoyu Shi, Dazhong Shen, Guanglu Song, Yu Liu, and Hongsheng Li. Be-your-outpainter: Mastering video outpainting through input-specific adaptation. In *European Conference on Computer Vision*, pages 153–168. Springer, 2025.

[65] Fu-Yun Wang, Ling Yang, Zhaoyang Huang, Mengdi Wang, and Hongsheng Li. Rectified diffusion: Straightness is not your need in rectified flow. *arXiv preprint arXiv:2410.07303*, 2024.

[66] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023.

[67] Yilun Xu, Mingyang Deng, Xiang Cheng, Yonglong Tian, Ziming Liu, and Tommi Jaakkola. Restart sampling for improving generative processes, 2023.

[68] Yilun Xu, Shangyuan Tong, and Tommi Jaakkola. Stable target field for reduced variance score estimation in diffusion models. *arXiv preprint arXiv:2302.00670*, 2023.

[69] Zeyue Xue, Guanglu Song, Qiushan Guo, Boxiao Liu, Zhuofan Zong, Yu Liu, and Ping Luo. Raphael: Text-to-image generation via large mixture of diffusion paths. *Advances in Neural Information Processing Systems*, 36, 2024.

[70] 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.

[71] Yuanhao Zhai, Kevin Lin, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Chung-Ching Lin, David Doermann, Junsong Yuan, and Lijuan Wang. Motion consistency model: Accelerating video diffusion with disentangled motion-appearance distillation. *arXiv preprint arXiv:2406.06890*, 2024.

[72] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Conference on Computer Vision and Pattern Recognition*, June 2018.

[73] Long Zhao, Sanghyun Woo, Ziyu Wan, Yandong Li, Han Zhang, Boqing Gong, Hartwig Adam, Xuhui Jia, and Ting Liu. $epsilon$-vae: Denoising as visual decoding. *arXiv preprint arXiv:2410.04081*, 2024.

[74] Jianbin Zheng, Minghui Hu, Zhongyi Fan, Chaoyue Wang, Changxing Ding, Dacheng Tao, and Tat-Jen Cham. Trajectory consistency distillation. *arXiv preprint arXiv:2402.19159*, 2024.

[75] Le Zhuo, Ruoyi Du, Han Xiao, Yangguang Li, Dongyang Liu, Rongjie Huang, Wenze Liu, Lirui Zhao, Fu-Yun Wang, Zhanyu Ma, et al. Lumina-next: Making lumina-t2x stronger and faster with next-dit. *arXiv preprint arXiv:2406.18583*, 2024.

 https://doi.org/10.52202/079017-2668

Appendix

I Related Works

I.1 Diffusion Models

Diffusion models [15, 57, 20, 73] have gradually become the dominant foundation models in image synthesis. Many works have greatly explored the nature of diffusion models [29, 6, 57, 22] and generalize/improve the design space of diffusion models [54, 20, 22]. Some works explore the model architecture for diffusion models [7, 40, 9, 75]. Some works scale up the diffusion models for text-conditioned synthesis or real-world applications [45, 41, 52, 64, 18, 36, 69, 17, 61]. Some works explore the sampling acceleration methods, including scheduler-level [20, 33, 54] or training-level [38, 56]. The formal ones are basically to explore better approximation of the PF-ODE [33, 54].

https://doi.org/10.52202/079017-2668

The latter are mainly distillation methods [38, 46, 30, 65, 70, 10] or initializing diffusion weights for GAN training [49, 27, 39, 19].

I.2 Consistency Models

The consistency model is a new family of generative models [11, 56, 55, 62, 26, 32] supporting fast and high quality generation. It can be trained either by distillation or direct training without teacher models. Improved techniques even allow consistency training excelling the performance of diffusion training [55]. Consistency trajectory model [21] proposes to learn the trajectory consistency, providing a more flexible framework. Some works combine [24] consistency training with GAN [24] for better training efficiency. Some works adopt continuous consistency models and achieve excelling performance [11, 62, 32]. Some works apply the idea of consistency model to language model [25] and policy learning [42, 34]. Some works extends the application scope of consistency models for text-conditioned image generation [35, 44, 74] and text-conditioned video generation [63, 37, 71, 60].

We notice that a recent work multistep consistency models [13] also proposes to splitting the ODE trajectory into multi-parts for consistency learning, and here we hope to credit their work for the valuable exploration. However, our work is principally different from multistep consistency models. Since they do not open-source their code and weights, we clarify the difference between our work and multistep consistency models based on the details of their technical report. Firstly, they didn't provide explicit model definitions and boundary conditions nor theoretically show the error bound to prove the soundness of their work. Yet in our work, we believe we have given an explicit definition of important components and theoretically shown the soundness of important techniques applied. For example, they claimed using DDIM for training and inference, while in our work, we have shown that although we can also optionally parameterize as the DDIM format, there's an intrinsic difference between that parameterization and DDIM (i.e., the difference between exact solution learning and first-order approximation). The aDDIM and invDDIM as highlighted in their pseudo code have no relation to PCM. Secondly, they are mainly for the unconditional or class-conditional generation, yet we aim for text-conditional generation in large models and dive into the influence of classifier-free guidance in consistency distillation. Besides, we introduce an adversarial loss that aligns well with consistency learning and improves the generation results in few-step settings. We recommend readers to read their work for better comparison.

II Proofs

II.1 Phased Consistency Distillation

The following is an extension of the original proof of consistency distillation for phased consistency distillation.

Theorem 1. *For arbitrary sub-trajectory* $[s_m, s_{m+1}]$. *let* $\Delta t_m := \max_{t_n, t_{n+1} \in [s_m, s_{m+1}]} \{|t_{n+1} - t_n|\}$, *and* $\boldsymbol{f}^m(\cdot, \cdot; \boldsymbol{\phi})$ *be the target phased consistency function induced by the pre-trained diffusion model (empirical PF-ODE). Assume the* $\boldsymbol{f}^m_{\boldsymbol{\theta}}$ *is L-Lipschitz and the ODE solver has local error uniformly bounded by* $\mathcal{O}((t_{n+1} - t_n)^{p+1})$ *with* $p \geq 1$. *Then, if* $\mathcal{L}_{PCM}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$, *we have*

$$\sup_{t_n, t_{n+1} \in [s_m, s_{m+1}], \mathbf{x}} \|\boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}, t_n) - \boldsymbol{f}^m(\mathbf{x}, t_n)\| = \mathcal{O}((\Delta t_m)^p).$$

Proof. From the condition $\mathcal{L}_{\text{PCM}}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$, for any $\mathbf{x}_{t_n}$ and $t_n, t_{n+1} \in [s_m, s_{m+1}]$, we have

$$\boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}, t_{n+1}}) \equiv \boldsymbol{f}^m_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}^{\boldsymbol{\phi}}_{t_n}, t_n) \tag{10}$$

Denote $\boldsymbol{e}^m_n := \boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) - \boldsymbol{f}^m(\mathbf{x}_{t_n}, t_n; \boldsymbol{\phi})$, we have

$$\begin{aligned}
\boldsymbol{e}^m_{n+1} &= \boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \boldsymbol{f}^m(\mathbf{x}_{t_{n+1}}, t_{n+1}; \boldsymbol{\phi}) \\
&= \boldsymbol{f}^m_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n) - \boldsymbol{f}^m(\mathbf{x}_{t_n}, t_n; \boldsymbol{\phi}) \\
&= \boldsymbol{f}^m_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n) - \boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) + \boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) - \boldsymbol{f}^m(\mathbf{x}_{t_n}, t_n; \boldsymbol{\phi}) \\
&= \boldsymbol{f}^m_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n) - \boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) + \boldsymbol{e}^m_n \,.
\end{aligned} \tag{11}$$

 https://doi.org/10.52202/079017-2668

Considering that $\boldsymbol{f}_{\boldsymbol{\theta}}^m$ is L-Lipschitz, we have

$$\begin{aligned}\|\boldsymbol{e}_{n+1}^m\|_2 &= \|\boldsymbol{e}_n^m + \boldsymbol{f}_{\boldsymbol{\theta}}^m(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n) - \boldsymbol{f}_{\boldsymbol{\theta}}^m(\mathbf{x}_{t_n}, t_n)\|_2 \\ &\leq \|\boldsymbol{e}_n^m\|_2 + \|\boldsymbol{f}_{\boldsymbol{\theta}}^m(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n) - \boldsymbol{f}_{\boldsymbol{\theta}}^m(\mathbf{x}_{t_n}, t_n)\|_2 \\ &\leq \|\boldsymbol{e}_n^m\|_2 + L\|\mathbf{x}_{t_n} - \mathbf{x}_{t_n}^{\boldsymbol{\phi}}\|_2 \\ &= \|\boldsymbol{e}_n^m\|_2 + L \cdot \mathcal{O}((t_{n+1} - t_n)^{p+1}) \\ &\leq \|\boldsymbol{e}_n^m\|_2 + L(t_{n+1} - t_n) \cdot \mathcal{O}((\Delta t_m)^p)\,.\end{aligned} \tag{12}$$

Besides, due to the boundary condition, we have

$$\begin{aligned}\boldsymbol{e}_{s_m}^m &= \boldsymbol{f}_{\boldsymbol{\theta}}^m(\mathbf{x}_{s_m}, s_m) - \boldsymbol{f}^m(\mathbf{x}_{s_m}, s_m; \boldsymbol{\phi}) \\ &= \mathbf{x}_{s_m} - \mathbf{x}_{s_m} \qquad\qquad . \\ &= 0\end{aligned} \tag{13}$$

Hence, we have

$$\begin{aligned}\|\boldsymbol{e}_{n+1}^m\|_2 &\leq \|\boldsymbol{e}_{s_m}^m\|_2 + L \cdot \mathcal{O}((\Delta t)^p) \sum_{t_i, t_{i+1} \in [s_m, s_{m+1}]} t_{i+1} - t_i \\ &= 0 + L \cdot \mathcal{O}((\Delta t)^p) \cdot (s_{m+1} - s_m) \\ &= \mathcal{O}((\Delta t_m)^p)\end{aligned} \tag{14}$$

□

Theorem 2. *For arbitrary set of sub-trajectories $\{[s_i, s_{i+1}]\}_{i=m}^{m'}$. Let $\Delta t_m := \max_{t_n, t_{n+1} \in [s_m, s_{m+1}]}\{|t_{n+1} - t_n|\}$, $\Delta t_{m,m'} := \max_{i \in [m', m]}\{\Delta t_i\}$, and $\boldsymbol{f}^{m,m'}(\cdot, \cdot; \boldsymbol{\phi})$ be the target phased consistency function induced by the pre-trained diffusion model (empirical PF-ODE). Assume the $\boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}$ is L-Lipschitz and the ODE solver has local error uniformly bounded by $\mathcal{O}((t_{n+1} - t_n)^{p+1})$ with $p \geq 1$. Then, if $\mathcal{L}_{PCM}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$, we have*

$$\sup_{t_n \in [s_m, s_{m+1}), \mathbf{x}} \|\boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}(\mathbf{x}, t_n) - \boldsymbol{f}^{m,m'}(\mathbf{x}, t_n)\| = \mathcal{O}((\Delta t_{m,m'})^p).$$

Proof. From the condition $\mathcal{L}_{\text{PCM}}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$, for any $\mathbf{x}_{t_n}$ and $t_n, t_{n+1} \in [s_m, s_{m+1}]$, we have

$$\boldsymbol{f}_{\boldsymbol{\theta}}^m(\mathbf{x}_{t_{n+1}, t_{n+1}}) \equiv \boldsymbol{f}_{\boldsymbol{\theta}}^m(\hat{\boldsymbol{x}}_{t_n}^{\boldsymbol{\phi}}, t_n) \tag{15}$$

Denote $\boldsymbol{e}_n^{m,m'} := \boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}(\mathbf{x}_{t_n}, t_n) - \boldsymbol{f}^{m,m'}(\mathbf{x}_{t_n}, t_n; \boldsymbol{\phi})$, we have

$$\begin{aligned}\boldsymbol{e}_{n+1}^{m,m'} &= \boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \boldsymbol{f}^{m,m'}(\mathbf{x}_{t_{n+1}}, t_{n+1}; \boldsymbol{\phi}) \\ &= \boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n) - \boldsymbol{f}^{m,m'}(\mathbf{x}_{t_n}, t_n; \boldsymbol{\phi}) \\ &= \boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n) - \boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}(\mathbf{x}_{t_n}, t_n) + \boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}(\mathbf{x}_{t_n}, t_n) - \boldsymbol{f}^{m,m'}(\mathbf{x}_{t_n}, t_n; \boldsymbol{\phi}) \\ &= \boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n) - \boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}(\mathbf{x}_{t_n}, t_n) + \boldsymbol{e}_n^{m,m'}\,.\end{aligned} \tag{16}$$

Considering that $\boldsymbol{f}_{\boldsymbol{\theta}}^m$ is L-Lipschitz, we have

$$\begin{aligned}\|\boldsymbol{e}_{n+1}^{m,m'}\|_2 &= \|\boldsymbol{e}_n^{m,m'} + \boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n) - \boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}(\mathbf{x}_{t_n}, t_n)\|_2 \\ &\leq \|\boldsymbol{e}_n^{m,m'}\|_2 + \|\boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n) - \boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}(\mathbf{x}_{t_n}, t_n)\|_2 \\ &\leq \|\boldsymbol{e}_n^{m,m'}\|_2 + L\|\boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'+1}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n) - \boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'+1}(\mathbf{x}_{t_n}, t_n)\|_2 \\ &\leq \|\boldsymbol{e}_n^{m,m'}\|_2 + L^2\|\boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'+2}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n) - \boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'+2}(\mathbf{x}_{t_n}, t_n)\|_2 \\ &\leq \quad \vdots \\ &\leq \|\boldsymbol{e}_n^{m,m'}\|_2 + L^{m-m'}\|\boldsymbol{f}_{\boldsymbol{\theta}}^{m}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n) - \boldsymbol{f}_{\boldsymbol{\theta}}^{m}(\mathbf{x}_{t_n}, t_n)\|_2 \\ &\leq \|\boldsymbol{e}_n^{m,m'}\|_2 + L^{m-m'+1}\|\mathbf{x}_{t_n} - \mathbf{x}_{t_n}^{\boldsymbol{\phi}}\|_2 \\ &= \|\boldsymbol{e}_n^{m,m'}\|_2 + L^{m-m'+1} \cdot \mathcal{O}((t_{n+1} - t_n)^{p+1}) \\ &\leq \|\boldsymbol{e}_n^{m,m'}\|_2 + L^{m-m'+1}(t_{n+1} - t_n) \cdot \mathcal{O}((\Delta t_m)^p)\,.\end{aligned} \tag{17}$$

Hence, we have

$$\begin{aligned}\|\boldsymbol{e}_{n+1}^{m,m'}\|_2 &\leq \|\boldsymbol{e}_{s_m}^{m,m'}\|_2 + L^{m-m'+1}\cdot\mathcal{O}((\Delta t)^p)\sum_{t_i,t_{i+1}\in[s_m,s_{m+1}]} t_{i+1}-t_i \\ &= \|\boldsymbol{e}_{s_m}^{m,m'}\|_2 + L^{m-m'+1}\cdot\mathcal{O}((\Delta t)^p)\cdot(s_{m+1}-s_m) \\ &= \|\boldsymbol{e}_{s_m}^{m,m'}\|_2 + \mathcal{O}((\Delta t_m)^p) \\ &= \|\boldsymbol{e}_{s_m}^{m-1,m'}\|_2 + \mathcal{O}((\Delta t_m)^p)\end{aligned} \quad , \tag{18}$$

Where the last equation is due to the boundary condition $\boldsymbol{f}_{\boldsymbol{\theta}}^{m,m'}(\mathbf{x}_{s_m}, s_m) = \boldsymbol{f}_{\boldsymbol{\theta}}^{m-1,m'}(\boldsymbol{f}_{\boldsymbol{\theta}}^{m}(\mathbf{x}_{s_m}, s_m), s_m) = \mathbf{f}_{\boldsymbol{\theta}}^{m-1,m'}(\mathbf{x}_{s_m}, s_m)$.
Thereby, we have

$$\begin{aligned}\|\boldsymbol{e}_{n+1}^{m,m'}\|_2 &\leq \|\boldsymbol{e}_{s_m}^{m-1,m'}\|_2 + \mathcal{O}((\Delta t_m)^p) \\ &\leq \|\boldsymbol{e}_{s_{m-1}}^{m-2,m'}\|_2 + \mathcal{O}((\Delta t_m)^p) + \mathcal{O}((\Delta t_{m-1})^p) \\ &\leq \quad \vdots \\ &\leq \|e_{s_{m'+1}}^{m'}\| + \sum_{i=m}^{m'+1}\mathcal{O}((\Delta t_i)^p) \\ &\leq \sum_{i=m}^{m'}\mathcal{O}((\Delta t_i)^p) \\ &\leq (m-m'+1)\mathcal{O}((\Delta t_{m,m'})^p) \\ &= \mathcal{O}((\Delta t_{m,m'})^p)\end{aligned} \tag{19}$$

□

II.2 Parameterization Equivalence

We show that the parameterization of Equation 3 is equal to the DDIM inference format [33, 54].

Theorem 3. *Define* $F_{\boldsymbol{\theta}}(\mathbf{x}_t, t, s) = \frac{\alpha_s}{\alpha_t}\mathbf{x}_t - \alpha_s\hat{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\int_{\lambda_t}^{\lambda_s} e^{-\lambda}\mathrm{d}\lambda$*, then the parameterization has the same format of DDIM* $\mathbf{x}_s = \alpha_s\left(\frac{\mathbf{x}_t - \sigma_t\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\alpha_t}\right) + \sigma_s\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$.

Proof.

$$\begin{aligned}F_{\boldsymbol{\theta}}(\mathbf{x}_t, t, s) &= \frac{\alpha_s}{\alpha_t}\mathbf{x}_t - \alpha_s\hat{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\int_{\lambda_t}^{\lambda_s} e^{-\lambda}\mathrm{d}\lambda \\ &= \frac{\alpha_s}{\alpha_t}\mathbf{x}_t - \alpha_s\hat{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\left(e^{-\lambda_t} - e^{-\lambda_s}\right) \\ &= \frac{\alpha_s}{\alpha_t}\mathbf{x}_t - \alpha_s\hat{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)(\frac{\sigma_t}{\alpha_t} - \frac{\sigma_s}{\alpha_s}) \\ &= \frac{\alpha_s}{\alpha_t}\mathbf{x}_t - \frac{\alpha_s\sigma_t}{\alpha_t}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) + \sigma_s\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \\ &= \alpha_s\left(\frac{\mathbf{x}_t - \sigma_t\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\alpha_t}\right) + \sigma_s\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\end{aligned} \tag{20}$$

□

II.3 Guided Distillation

We show the relationship of epsilon prediction of consistency models trained with guided distillation and diffusion models when considering the classifier-free guidance.

https://doi.org/10.52202/079017-2668

Theorem 4. *Assume the consistency model $\boldsymbol{\epsilon_\theta}$ is trained by consistency distillation with the teacher diffusion model $\epsilon_{\boldsymbol{\phi}}$, and the ODE solver is augmented with CFG value w and null text embedding $\varnothing$. Let $\boldsymbol{c}$ and $\boldsymbol{c}_{neg}$ be the prompt and negative prompt applied for the inference of consistency model. Then, if the ODE solver is perfect and the $\mathcal{L}_{PCM}(\boldsymbol{\theta}, \boldsymbol{\theta}) = 0$, we have*

$$\begin{aligned}\boldsymbol{\epsilon_\theta}(\mathbf{x}, t, \boldsymbol{c}, \boldsymbol{c_{neg}}; w') \propto ww' \left[\boldsymbol{\epsilon_\phi}(\mathbf{x}, t, \boldsymbol{c}) - ((1 - \frac{w-1}{ww'})\boldsymbol{\epsilon_\phi}(\mathbf{x}, t, \boldsymbol{c}_{neg}) + \frac{w-1}{ww'}\boldsymbol{\epsilon_\phi}(\mathbf{x}, t, \varnothing))\right] \\ + \boldsymbol{\epsilon_\phi}(\mathbf{x}, t, \boldsymbol{c}_{neg})\end{aligned}$$

Proof. If the ODE solver is perfect, that means the empirical PF-ODE is exactly the PF-ODE of the training data. Then considering Theorem 1 and Theorem 2, it is apparent that the consistency model will fit the PF-ODE. To show that, considering the case in Theorem 1, we have

$$\begin{aligned}\boldsymbol{e}^m_{n+1} &= \boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \boldsymbol{f}^m(\mathbf{x}_{t_{n+1}}, t_{n+1}; \boldsymbol{\phi}) \\ &= \boldsymbol{f}^m_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n) - \boldsymbol{f}^m(\mathbf{x}_{t_n}, t_n; \boldsymbol{\phi}) \\ &= \boldsymbol{f}^m_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n) - \boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) + \boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) - \boldsymbol{f}^m(\mathbf{x}_{t_n}, t_n; \boldsymbol{\phi}) \\ &= \boldsymbol{f}^m_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n) - \boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) + \boldsymbol{e}^m_n \\ &\overset{(i)}{=} \boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) - \boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) + \boldsymbol{e}^m_n \\ &= \boldsymbol{e}^m_n \\ &= \vdots \\ &= \boldsymbol{e}^m_{s_m} \\ &= 0\,,\end{aligned} \tag{21}$$

where (i) is because the ODE solver is perfect. Considering our parameterization in Equation 3, then it should be equal to the exact solution in Equation 2. That is,

$$\begin{aligned}\frac{\alpha_s}{\alpha_t}\mathbf{x}_t - \alpha_s\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t)\int_{\lambda_t}^{\lambda_s} e^{-\lambda}\mathrm{d}\lambda &= \frac{\alpha_s}{\alpha_t}\mathbf{x}_t + \alpha_s\int_{\lambda_t}^{\lambda_s} e^{-\lambda}\sigma_{t_\lambda(\lambda)}\nabla\log\mathbb{P}_{t_\lambda(\lambda)}(\mathbf{x}_{t_\lambda(\lambda)})\mathrm{d}\lambda \\ \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\int_{\lambda_t}^{\lambda_s} e^{-\lambda}\mathrm{d}\lambda &= \int_{\lambda_t}^{\lambda_s} e^{-\lambda}\boldsymbol{\epsilon_\phi}(\mathbf{x}_{t_\lambda(\lambda)}, t_\lambda(\lambda))\mathrm{d}\lambda \\ \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) &= \frac{\int_{\lambda_t}^{\lambda_s} e^{-\lambda}\boldsymbol{\epsilon_\phi}(\mathbf{x}_{t_\lambda(\lambda)}, t_\lambda(\lambda))\mathrm{d}\lambda}{\int_{\lambda_t}^{\lambda_s} e^{-\lambda}\mathrm{d}\lambda}\,.\end{aligned} \tag{22}$$

Then we have the epsilon prediction is weighted integral of diffusion-based epsilon prediction on the trajectory. Therefore, it is apparent that, for any $t' \leq t$ and t' on the same sub-trajectory with t, we have

$$\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \propto \epsilon_{\boldsymbol{\phi}}(\mathbf{x}_{t'}, t') \tag{23}$$

When considering the text-conditioned generation and the ODE solver being augmented with the CFG value w, we have

$$\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, \boldsymbol{c}) \propto \boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \varnothing) + w(\boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \boldsymbol{c}) - \boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \varnothing))\,. \tag{24}$$

If we additionally apply the classifier-free guidance to the consistency models with negative prompt embedding $\boldsymbol{c}_{\mathrm{neg}}$ and CFG value w', we have

$$\begin{aligned}&\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, \boldsymbol{c}, \boldsymbol{c}_{\mathrm{neg}}; w') \\ &= \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, \boldsymbol{c}_{\mathrm{neg}}) + w'(\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, \boldsymbol{c}) - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, \boldsymbol{c}_{\mathrm{neg}})) \\ &\propto \boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \varnothing) + w(\boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \boldsymbol{c}) - \boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \varnothing)) \\ &+ w'\{[\boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \varnothing) + w(\boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \boldsymbol{c}) - \boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \varnothing))] \\ &- [\boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \varnothing) + w(\boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \boldsymbol{c}) - \boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \varnothing))]\} \\ &= \boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \varnothing) + w(\boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \boldsymbol{c}_{\mathrm{neg}}) - \boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \varnothing)) + ww'(\boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \boldsymbol{c}) - \boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \boldsymbol{c}_{\mathrm{neg}})) \\ &= ww'(\boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \boldsymbol{c}) - ((1-\alpha)\boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \boldsymbol{c}_{\mathrm{neg}}) + \alpha\boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \varnothing))) + \boldsymbol{\epsilon_\phi}(\mathbf{x}_{t'}, t', \boldsymbol{c}_{\mathrm{neg}})\end{aligned} \tag{25}$$

□

II.4 Distribution Consistency Convergence

We firstly show that when $\mathcal{L}_{\text{PCM}} = 0$ is achieved, our distribution consistency loss will also converge to zero. Then, we additionally show that, when considering the pre-train data distribution and distillation data distribution mismatch, combining GAN is a flawed design. The loss is still non-zero even when the self-consistency is achieved, thus corrupting the training.

Theorem 5. *Denote the data distribution applied for consistency distillation phased is $\mathbb{P}_0$. And considering that the forward conditional probability path is defined by $\alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}$, we further define the intermediate distribution $\mathbb{P}_t(\mathbf{x}) = (\mathbb{P}_0(\frac{\mathbf{x}}{\alpha_t}) \cdot \frac{1}{\alpha_t}) * \mathcal{N}(0, \sigma_t)$. Similarly, we denote the data distribution applied for pretraining the diffusion model is $\mathbb{P}_0^{pretrain}(\mathbf{x})$ and the intermediate distribution following forward process are $\mathbb{P}_t^{pretrain}(\mathbf{x}) = (\mathbb{P}_t^{pretrain}(\mathbf{x}) = (\mathbb{P}_0^{pretrain}(\frac{\mathbf{x}}{\alpha_t}) \cdot \frac{1}{\alpha_t}) * \mathcal{N}(0, \sigma_t))$. This is reasonable since current large diffusion models are typically trained with much more resources on much larger datasets compared to those of consistency distillation. And, we denote the flow $\mathcal{T}^{\boldsymbol{\phi}}_{t\to s}$, $\mathcal{T}^{\boldsymbol{\theta}}_{t\to s}$, and $\mathcal{T}^{\boldsymbol{\phi}'}_{t\to s}$ correspond to our consistency model, pre-trained diffusion model, and the PF-ODE of the data distribution used for consistency distillation, respectively. Additionally, let $\mathcal{T}^{-}_{s\to t}$ be the distribution transition following the forward process SDE (adding noise). Then, if the $\mathcal{L}_{PCM} = 0$, for arbitrary sub-trajectory $[s_m, s_{m+1}]$, we have,*

$$\mathcal{L}^{adv}_{PCM}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}, m) = D\left(\mathcal{T}^{-}_{s_m\to s}\mathcal{T}^{\boldsymbol{\theta}}_{t_{n+1}\to s_m}\#\mathbb{P}_{t_{n+1}} \middle\| \mathcal{T}^{-}_{s_m\to s}\mathcal{T}^{\boldsymbol{\theta}}_{t_n\to s_m}\mathcal{T}^{\boldsymbol{\phi}}_{t_{n+1}\to t_n}\#\mathbb{P}_{t_{n+1}}\right) = 0.$$

Proof. Firstly, considering that the forward process $\mathcal{T}^{-}_{s\to t}$ is equivalent to scaling the original variables and then performing convolution operations with $\mathcal{N}(\mathbf{0}, \sigma^2_{s\to t}\mathbf{I})$. Therefore, as long as

$$D\left(\mathcal{T}^{\boldsymbol{\theta}}_{t_{n+1}\to s_m}\#\mathbb{P}_{t_{n+1}} \middle\| \mathcal{T}^{\boldsymbol{\theta}}_{t_n\to s_m}\mathcal{T}^{\boldsymbol{\phi}}_{t_{n+1}\to t_n}\#\mathbb{P}_{t_{n+1}}\right) = 0\,, \tag{26}$$

then we have

$$D\left(\mathcal{T}^{-}_{s_m\to s}\mathcal{T}^{\boldsymbol{\theta}}_{t_{n+1}\to s_m}\#\mathbb{P}_{t_{n+1}} \middle\| \mathcal{T}^{-}_{s_m\to s}\mathcal{T}^{\boldsymbol{\theta}}_{t_n\to s_m}\mathcal{T}^{\boldsymbol{\phi}}_{t_{n+1}\to t_n}\#\mathbb{P}_{t_{n+1}}\right) = 0\,. \tag{27}$$

From the condition $\mathcal{L}_{\text{PCM}}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$, for any $\mathbf{x}_{t_n} \in \mathbb{P}_{t_n}$ and $\mathbf{x}_{t_{n+1}} \in \mathbb{P}_{t_{n+1}}$ and $t_n, t_{n+1} \in [s_m, s_{m+1}]$, we have

$$\boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1},t_{n+1}}) \equiv \boldsymbol{f}^m_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}^{\boldsymbol{\phi}}_{t_n}, t_n)\,, \tag{28}$$

which induces that

$$\mathcal{T}^{\boldsymbol{\theta}}_{t_{n+1}\to s_m}\#\mathbb{P}_{t_{n+1}} \equiv \mathcal{T}^{\boldsymbol{\theta}}_{t_n\to s_m}\mathcal{T}^{\boldsymbol{\phi}}_{t_{n+1}\to t_n}\#\mathbb{P}_{t_{n+1}}\,. \tag{29}$$

Therefore, we show that if $\mathcal{L}_{PCM}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$, then

$$\mathcal{L}^{adv}_{\text{PCM}}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0 \tag{30}$$

□

Theorem 6. *Denote the data distribution applied for consistency distillation phased is $\mathbb{P}_0$. And considering that the forward conditional probability path is defined by $\alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}$, we further define the intermediate distribution $\mathbb{P}_t(\mathbf{x}) = (\mathbb{P}_0(\frac{\mathbf{x}}{\alpha_t}) \cdot \frac{1}{\alpha_t}) * \mathcal{N}(0, \sigma_t)$. Similarly, we denote the data distribution applied for pretraining the diffusion model is $\mathbb{P}_0^{pretrain}(\mathbf{x})$ and the intermediate distribution following forward process are $\mathbb{P}_t^{pretrain}(\mathbf{x}) = (\mathbb{P}_t^{pretrain}(\mathbf{x}) = (\mathbb{P}_0^{pretrain}(\frac{\mathbf{x}}{\alpha_t}) \cdot \frac{1}{\alpha_t}) * \mathcal{N}(0, \sigma_t))$. This is reasonable since current large diffusion models are typically trained with much more resources on much larger datasets compared to those of consistency distillation. And, we denote the flow $\mathcal{T}^{\boldsymbol{\phi}}_{t\to s}$, $\mathcal{T}^{\boldsymbol{\theta}}_{t\to s}$, and $\mathcal{T}^{\boldsymbol{\phi}'}_{t\to s}$ correspond to our consistency model, pre-trained diffusion model, and the PF-ODE of the data distribution used for consistency distillation, respectively. Then, if the $\mathcal{L}_{PCM} = 0$, for arbitrary sub-trajectory $[s_m, s_{m+1}]$, we have,*

$$\mathcal{L}^{adv}_{PCM}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}, m) = D\left(\mathcal{T}^{\boldsymbol{\theta}}_{t_{n+1}\to s_m}\#\mathbb{P}_{t_{n+1}} \middle\| \mathbb{P}_{s_m}\right) \geq 0.$$

https://doi.org/10.52202/079017-2668

Proof. From the condition $\mathcal{L}_{\text{PCM}}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$, for any $\mathbf{x}_{t_n} \in \mathbb{P}_{t_n}$ and $\mathbf{x}_{t_{n+1}} \in \mathbb{P}_{t_{n+1}}$ and $t_n, t_{n+1} \in [s_m, s_{m+1}]$, we have

$$\boldsymbol{f}_{\boldsymbol{\theta}}^m(\mathbf{x}_{t_{n+1},t_{n+1}}) \equiv \boldsymbol{f}_{\boldsymbol{\theta}}^m(\hat{\boldsymbol{x}}_{t_n}^{\boldsymbol{\phi}}, t_n)\,, \tag{31}$$

which induces that

$$\mathcal{T}_{t_{n+1}\to s_m}^{\boldsymbol{\theta}} \# \mathbb{P}_{t_{n+1}} \equiv \mathcal{T}_{t_{n+1}\to s_m}^{\boldsymbol{\phi}} \# \mathbb{P}_{t_{n+1}}\,. \tag{32}$$

Besides, since $\mathcal{T}_{t\to s}^{\phi'}$ corresponds to the PF-ODE of the data distribution used for consistency distillation, we can rewrite

$$\mathbb{P}_{s_m} \equiv \mathcal{T}_{t_{n+1}\to s_m}^{\boldsymbol{\phi}'} \# \mathbb{P}_{t_{n+1}}\,. \tag{33}$$

Therefore, we have

$$D\left(\mathcal{T}_{t_{n+1}\to s_m}^{\boldsymbol{\theta}} \# \mathbb{P}_{t_{n+1}} \middle\| \mathbb{P}_{s_m}\right) = D\left(\mathcal{T}_{t_{n+1}\to s_m}^{\boldsymbol{\phi}} \# \mathbb{P}_{t_{n+1}} \middle\| T_{t_{n+1}\to s_m}^{\boldsymbol{\phi}'} \# \mathbb{P}_{t_{n+1}}\right). \tag{34}$$

Since we know that $\mathbb{P}_0^{pretrain} \neq \mathbb{P}_0$. Therefore, there exists m and n achieving the strict inequality. Specifically, if we set $s_m = \epsilon$ and $t_{n+1} = T$, we have

$$D\left(\mathcal{T}_{T\to\epsilon}^{\boldsymbol{\phi}} \# \mathbb{P}_T \middle\| T_{T\to\epsilon}^{\boldsymbol{\phi}'} \# \mathbb{P}_T\right) = D\left(\mathbb{P}_0^{pretrain} \middle\| \mathbb{P}_0\right) > 0\,. \tag{35}$$

□

III Discussions

III.1 Numerical Issue of Parameterization

Even though our above-discussed parameterization is theoretically sound, it poses a numerical issue when applied to the epsilon-prediction-based models, especially for the one-step generation. To be specific, for the one-step generation, we are required to predict the solution point $\mathbf{x}_0$ with $\mathbf{x}_t$ from the self-consistency property of consistency models. With epsilon-prediction models, the formula can represented as the following

$$\mathbf{x}_0 = \frac{\mathbf{x}_t - \sigma_t \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\alpha_t}\,. \tag{36}$$

However, when inference, we should choose $t = T$ since it is the noisy timestep that is closest to the normal distribution. For the DDPM framework, there should be $\sigma_T \approx 1$ and $\alpha_T \approx 0$ to make the noisy timestep T as close to the normal distribution as possible. For instance, the noise schedulers applied by Stable Diffusion v1-5 and Stable Diffusion XL all have the $\alpha_T = 0.068265$. Therefore, we have

$$\mathbf{x}_0 = \frac{\mathbf{x}_T - \sigma_T \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\alpha_T} \approx 14.64(\mathbf{x}_T - \sigma_T \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t))\,. \tag{37}$$

This indicates that we will multiply over $10\times$ to the model prediction. In our experiments, we find the SDXL is more influenced by this issue, tending to produce artifact points or lines.

Generally speaking, using the x0-prediction and v-prediction can well solve these issues. It is obvious for the x0-prediction. For the v-prediction, we have

$$\mathbf{x}_0 = \alpha_t \mathbf{x}_t - \sigma_t \boldsymbol{v}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\,. \tag{38}$$

However, since the diffusion models are trained with the epsilon-prediction, transforming them into the other prediction formats takes additional computation resources and might harm the performance due to the relatively large gaps among different prediction formats.

We solve this issue through a simple way that we set a clip boundary to the value of α_t. Specifically, we apply

$$\mathbf{x}_0 = \frac{\mathbf{x}_t - \sigma_t \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\max\{\alpha_t, 0.5\}}\,. \tag{39}$$

III.2 Why CTM Needs Target Timestep Embeddings?

Consistency trajectory model (CTM), as we discussed, proposes to learn how 'move' from arbitrary points on the ODE trajectory to arbitrary other points. Thereby, the model should be aware of 'where they are' and 'where they target' for precise prediction.

Basically, they can still follow our parameterization with an additional target timestep embedding.

$$\mathbf{x}_s = \frac{\alpha_s}{\alpha_t}\mathbf{x}_t - \alpha_s \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, s) \int_{\lambda_t}^{\lambda_s} e^{-\lambda} \mathrm{d}\lambda \tag{40}$$

In this design, we have

$$\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, s) = \frac{\int_{\lambda_t}^{\lambda_s} e^{-\lambda} \boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_{t_\lambda(\lambda)}, t_\lambda(\lambda)) \mathrm{d}\lambda}{\int_{\lambda_t}^{\lambda_s} e^{-\lambda} \mathrm{d}\lambda}\,. \tag{41}$$

Here, we show that dropping the target timestep embedding s is a **flawed** design.

Assume we hope to predict the results at timestep s' and timestep s'' from timestep t and sample $\mathbf{x}_t$, where $s' < s''$.

If we hope to learn both $\mathbf{x}_{s'}$ and $\mathbf{x}_{s''}$ correctly, without the target timestep embedding, we must have

$$\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = \frac{\int_{\lambda_t}^{\lambda_{s'}} e^{-\lambda} \boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_{t_\lambda(\lambda)}, t_\lambda(\lambda)) \mathrm{d}\lambda}{\int_{\lambda_t}^{\lambda_{s'}} e^{-\lambda} \mathrm{d}\lambda} \tag{42}$$

$$\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = \frac{\int_{\lambda_t}^{\lambda_{s''}} e^{-\lambda} \boldsymbol{\epsilon}_{\boldsymbol{\phi}}(\mathbf{x}_{t_\lambda(\lambda)}, t_\lambda(\lambda)) \mathrm{d}\lambda}{\int_{\lambda_t}^{\lambda_{s''}} e^{-\lambda} \mathrm{d}\lambda}\,. \tag{43}$$

This will lead to the conclusion that all the segments $[t, s]$ on the ODE trajectory have the same weighted epsilon prediction, which violates the truth, ignoring the dynamic variations and dependencies specific to each segments.

III.3 Contributions

Here we re-emphasize the key components of PCM and summarize the contributions of our work.

The motivation of our work is to accelerate the sampling of high-resolution text-to-image and text-to-video generation with consistency models training paradigm. Previous work, latent consistency model (LCM), tried to replicate the power of the consistency model in this challenging setting but did not achieve satisfactory results. We observe and analyze the limitations of LCM from three perspectives and propose PCM, generalizing the design space and tackling all these limitations.

At the heart of PCM is to phase the whole ODE trajectory into multiple phases. Each phase corresponding to a sub-trajectory is treated as an independent consistency model learning objective. We provide a standard definition of PCM and show that the optimal error bounds between the prediction of trained PCM and PF-ODE are upper-bounded by $\mathcal{O}((\Delta t)^p)$. The phasing technique allows for deterministic multi-step sampling, ensuring consistent generation results under different inference steps.

Additionally, we provide a deep analysis of the parameterization when converting pre-trained diffusion models into consistency models. From the exact solution format of PF-ODE, we propose a simple yet effective parameterization and show that it has the same formula as first-order ODE solver DDIM. However, we show that the parameterization has a natural difference from the DDIM ODE solver. That is the difference between exact solution learning and score estimation. Besides, we point out that even though the parameterization is theoretically sound, it poses numerical issues when building the one-step generator. We propose a simple yet effective threshold clip strategy for the parameterization.

We introduce an innovative adversarial loss, which greatly boosts the generation quality at a low-step regime. We implement the loss in a GAN style. However, we show that our method has an intrinsic

 https://doi.org/10.52202/079017-2668

difference from traditional GAN. We show that our introduced adversarial loss will converge to zero when the self-consistency property is achieved. However, the GAN loss will still be non-zero when considering the distribution distance between the pre-trained dataset for diffusion training and the distillation dataset for consistency distillation. We investigate the structures of discriminators and compare the latent-based discriminator and pixel-based discriminator. Our finding is that applying the latent-based visual backbone of pre-trained diffusion U-Net makes the discriminator design simple and can produce better visual results compared to pixel-based discriminators. We also implement a similar discriminator structure for the text-to-video generation with a temporal inflated U-Net.

For the setting of text-to-image generation and text-to-video generation, classifier-free guidance (CFG) has become an important technique for achieving better controllability and generation quality. We investigate the influence of CFG on consistency distillation. And, to our knowledge, we, for the first time, point out the relations of consistency model prediction and diffusion model prediction when considering the CFG-augmented ODE solver (guided distillation). We show that this technique causes the trained consistency models unable to use large CFG values and be less sensitive to the negative prompt.

We achieve state-of-the-art few-step text-to-image generation and text-to-video generation with only 8 A 800 GPUs, indicating the advancements of our method.

IV Sampling

IV.1 Introducing Randomness for Sampling

For a given initial sample at timestep t belonging to the sub-trajectory $[s_m, s_{m+1}]$, we can support deterministic sampling according to the definition of $\boldsymbol{f}^{m,0}$.

However, previous work [21, 67] reveals that introducing a certain degree of stochasticity can lead to better generation results. We also observe a similar trade-off phenomenon in our practice. Therefore, we reparameterize the $F_{\boldsymbol{\theta}}(\mathbf{x}, t, s)$ as

$$\alpha_s\left(\frac{\mathbf{x}_t - \sigma_t \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\alpha_t}\right) + \sigma_s\left(\sqrt{r}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) + \sqrt{(1-r)}\boldsymbol{\epsilon}\right), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{44}$$

By controlling the value of $r \in [0, 1]$, we can determine the stochasticity for generation. With $r = 1$, it is pure deterministic sampling. With $r = 0$, it degrades to pure stochastic sampling. Note that, introducing the random $\boldsymbol{\epsilon}$ will make the generation results be away from the target induced by the diffusion models. However, since $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ and ϵ all follows the normal distribution, we have $\sqrt{r}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) + \sqrt{(1-r)}\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and thereby the predicted $\mathbf{x}_s$ should still follow the same distribution $\mathbb{P}_s$ approximately.

IV.2 Improving Diversity for Generation

Our another observation is the generation diversity with guided distillation is limited compared to original diffusion models. This is due to that the consistency models are distilled with a relatively large CFG value for guided distillation. The large CFG value, though known for enhancing the generation quality and text-image alignment, will degrade the generation diversity. We explore a simple yet effective strategy by adjusting the CFG values. $w'\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}, t, \boldsymbol{c}) + (1 - w')\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}, t, \varnothing)$, where $w' \in (0.5, 1]$. The epsilon prediction is the convex combination of the conditional prediction and the unconditional prediction.

V Societal Impact

V.1 Positive Societal Impact

We firmly believe that our work has a profound positive impact on society. While diffusion techniques excel in producing high-quality images and videos, their iterative inference process incurs significant computational and power costs. Our approach accelerates the inference of general diffusion models by up to 20 times or more, thus substantially reducing computational and power consumption. Moreover, it fosters enthusiasm among creators engaged in AI-generated content creation and lowers entry barriers.

V.2 Negative Societal Impact

Addressing potential negative consequences, given the generative nature of the model, there's a risk of generating false or harmful content.

V.3 Safeguards

To mitigate this, we conduct harmful information detection on user-provided text inputs. If harmful prompts are detected, the generation process is halted. Additionally, our method, fortunately, builds upon the open-source Stable Diffusion model, which includes an internal safety checker for detecting harmful content. This feature significantly enhances our ability to prevent the generation of harmful content.

VI Implementation Details

VI.1 Training Details

For the comparison of baselines, the training code for InstaFlow [31], SDXL-Turbo [49], SD-Turbo [49], TCD [74], and SDXL-Lightning [27] are not open-sourced yet, so we only compare their open-source weights.

For LCM [35], CTM [21], and our method, they are all implemented by us and trained with the same configuration.

Specifically, for multi-step models, we trained LoRA with a rank of 64. For models based on SD v1-5, we used a learning rate of 5e-6, a batch size of 160, and trained for 5k iterations. For models based on SDXL, we used a learning rate of 5e-6, a batch size of 80, and trained for 10k iterations. We did not use EMA for LoRA training.

For single-step models, we followed the approach of SD-Turbo and SDXL-Lightning, training all parameters. For models based on SD v1-5, we used a learning rate of 5e-6, a batch size of 160, and trained for 10k iterations. For models based on SDXL, we used a learning rate of 1e-6, a batch size of 16, and trained for 50k iterations. We used EMA=0.99 for training.

For CTM, we additionally learned a timestep embedding to indicate the target timestep.

For all training settings, we uniformly sample 50 timesteps from the 1000 timesteps of StableDiffusion and apply DDIM as the ODE solver.

VI.2 Pseudo Training Code

Algorithm 1 Phased Consistency Distillation with CFG-augmented ODE solver (PCD)

Input: dataset $\mathcal{D}$, initial model parameter $\boldsymbol{\theta}$, learning rate η, ODE solver $\Psi(\cdot,\cdot,\cdot,\cdot)$, distance metric $d(\cdot,\cdot)$, EMA rate μ, noise schedule α_t, σ_t, guidance scale $[w_{\text{min}}, w_{\text{max}}]$, number of ODE step k, discretized timesteps $t_0 = \epsilon < t_1 < t_2 < \cdots < t_N = T$, edge timesteps $s_0 = t_0 < s_1 < s_2 < \cdots < s_M = t_N \in \{t_i\}_{i=0}^{N}$ to split the ODE trajectory into M sub-trajectories.
Training data : $\mathcal{D}_{\mathbf{x}} = \{(\mathbf{x}, \boldsymbol{c})\}$
$\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$
repeat
  Sample $(\boldsymbol{z}, \boldsymbol{c}) \sim \mathcal{D}_z$, $n \sim \mathcal{U}[0, N-k]$ and $\omega \sim [\omega_{\text{min}}, \omega_{\text{max}}]$
  Sample $\mathbf{x}_{t_{n+k}} \sim \mathcal{N}(\alpha_{t_{n+k}}\boldsymbol{z}; \sigma^2_{t_{n+k}}\mathbf{I})$
  Determine $[s_m, s_{m+1}]$ given n
  $\mathbf{x}^{\boldsymbol{\phi}}_{t_n} \leftarrow (1+\omega)\Psi(\mathbf{x}_{t_{n+k}}, t_{n+k}, t_n, \boldsymbol{c}) - \omega\Psi(\mathbf{x}_{t_{n+k}}, t_{n+k}, t_n, \varnothing)$
  $\tilde{\mathbf{x}}_{s_m} = \boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+k}}, t_{n+k}, \boldsymbol{c})$ and $\hat{\mathbf{x}}_{s_m} = \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}^{\boldsymbol{\phi}}_{t_n}, t_n, \boldsymbol{c})$
  Obtain $\tilde{\mathbf{x}}_s$ and $\hat{\mathbf{x}}_s$ through adding noise to $\tilde{\mathbf{x}}_{s_m}$ and $\hat{\mathbf{x}}_{s_m}$
  $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) = d(\tilde{\mathbf{x}}_{s_m}, \hat{\mathbf{x}}_{s_m}) + \lambda(\text{ReLU}(1+\tilde{\mathbf{x}}_s) + \text{ReLU}(1-\hat{\mathbf{x}}_s))$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-)$
  $\boldsymbol{\theta}^- \leftarrow \text{stopgrad}(\mu\boldsymbol{\theta}^- + (1-\mu)\boldsymbol{\theta})$
until convergence

https://doi.org/10.52202/079017-2668

Algorithm 2 Phased Consistency Distillation with normal ODE solver (PCD*)

Input: dataset $\mathcal{D}$, initial model parameter $\boldsymbol{\theta}$, learning rate η, ODE solver $\Psi(\cdot,\cdot,\cdot,\cdot)$, distance metric $d(\cdot,\cdot)$, EMA rate μ, noise schedule α_t, σ_t, drop ratio η, number of ODE step k, discretized timesteps $t_0 = \epsilon < t_1 < t_2 < \cdots < t_N = T$, edge timesteps $s_0 = t_0 < s_1 < s_2 < \cdots < s_M = t_N \in \{t_i\}_{i=0}^{N}$ to split the ODE trajectory into M sub-trajectories.
Training data : $\mathcal{D}_{\mathbf{x}} = \{(\mathbf{x}, \boldsymbol{c})\}$
$\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$
repeat
  Sample $(\boldsymbol{z}, \boldsymbol{c}) \sim \mathcal{D}_z$, $n \sim \mathcal{U}[0, N-k]$
  Sample $\mathbf{x}_{t_{n+k}} \sim \mathcal{N}(\alpha_{t_{n+k}}\boldsymbol{z}; \sigma^2_{t_{n+k}}\mathbf{I})$
  Determine $[s_m, s_{m+1}]$ given n
  $r \sim \mathcal{U}[0,1]$
  if $r < \eta$ **then**
  $\boldsymbol{c} = \varnothing$
  end if
  $\mathbf{x}^{\phi}_{t_n} \leftarrow \Psi(\mathbf{x}_{t_{n+k}}, t_{n+k}, t_n, \boldsymbol{c})$
  $\tilde{\mathbf{x}}_{s_m} = \boldsymbol{f}^m_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+k}}, t_{n+k}, \boldsymbol{c})$ and $\hat{\mathbf{x}}_{s_m} = \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x}^{\phi}_{t_n}, t_n, \boldsymbol{c})$
  Obtain $\tilde{\mathbf{x}}_s$ and $\hat{\mathbf{x}}_s$ through adding noise to $\tilde{\mathbf{x}}_{s_m}$ and $\hat{\mathbf{x}}_{s_m}$
  $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) = d(\tilde{\mathbf{x}}_{s_m}, \hat{\mathbf{x}}_{s_m}) + \lambda(\text{ReLU}(1+\tilde{\mathbf{x}}_s) + \text{ReLU}(1-\hat{\mathbf{x}}_s))$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-)$
  $\boldsymbol{\theta}^- \leftarrow \text{stopgrad}(\mu\boldsymbol{\theta}^- + (1-\mu)\boldsymbol{\theta})$
until convergence

VI.3 Decoupled Consistency Distillation For Video Generation

Our video generation model follows the design space of most current text-to-video generation models, viewing videos as temporal stacks of images and inserting temporal blocks to a pre-trained text-to-image generation U-Net to accommodate 3D features of noisy video inputs. We term this process temporal inflation.

Video generation models are typically much more resource-consuming than image generation models. Also, the overall caption and visual quality of video datasets are generally inferior to those of image datasets. Therefore, we apply the decoupled consistency learning to ease the training burden of text-to-video generation, which was first proposed by the previous work AnimateLCM. To be specific, we first conduct phased consistency distillation on stable diffusion v1-5 to obtain the one-step text-to-image generation models. Then we apply the temporal inflation to adapt the text-to-image generation model for video generation. Eventually, we conduct phased consistency distillation on the video dataset. We observe an obvious training speed-up with this decoupled strategy. This allows us to train the state-of-the-art fast video generation models with only 8 A 800 GPUs.

VI.4 Discriminator Design of Image Generation and Video Generation

VI.4.1 Image Generation

The overall discriminator design was greatly inspired by previous work StyleGAN-T [48], which showed that a pre-trained visual backbone can work as a great discriminator. For training the discriminator, we freeze the original weight of pre-trained visual backbones and insert light-weight Convolution-based discriminator heads for training.

Discriminator backbones. We consider two types of visual backbones as the discriminator backbone: pixel-based and latent-based.

We apply DINO [4] as the pixel-based backbone. To be specific, once obtaining the denoised latent code, we decode it through the VAE decoder to obtain the generated image. Then, we resize the generated image into the resolution that DINO trained with (e.g., 224×224) and feed it as the inputs of the DINO backbone. We extract the hidden features of the DINO backbone of different layers and feed them to the corresponding discriminator heads. Since DINO is trained in a self-supervised manner and no texts are used for training. Therefore, it is necessary to incorporate the text embeddings in the discriminator heads for text-to-image generation. To achieve that, we use the embedding of [CLS] token in the last layer of the pre-trained CLIP text encoder, linearly map it to the same dimension of discriminator output, and conduct affine transformation to the discriminator output. However, the pixel-based backbones have inevitable drawbacks. Firstly, it requires mapping

the latent code to a very high-dimensional image through the VAE decoder, which is much more costly compared to traditional diffusion training. Secondly, it can only accept the inputs of clean images, which poses challenges to PCM training, since we require the discrimination of inputs of intermediate noisy inputs. Thirdly, it requires resizing the images into the relatively low resolution it trained with. When applying this backbone design with high-dimensional image generation with Stable-Diffusion XL, we observe unsatisfactory results.

We apply the U-Net of the pre-trained diffusion model as the latent-based backbone. It has several advantages compared to the pixel-based backbones. Firstly, trained on the latent space, the U-Net possesses rich knowledge about latent space and can directly work as the discriminator at the latent space, avoiding costly decoding processes. Besides, it can accept the inputs of latent code at different timesteps, which aligns well with the design of PCM and can be applied for regularizing the consistency of all the intermediate distributions instead (i.e., $\mathbb{P}_t(\mathbf{x})$) of distribution at the edge points (i.e., $\mathbb{P}_{s_m}(\mathbf{x})$). Additionally, since the text information is already encoded in the U-Net backbone, we do not need to incorporate the text information in the discriminator head, which simplifies the discriminator design. We insert several randomly initialized lightweight simple discriminator heads after each block of the pre-trained U-Net. Each discriminator head consists of two lightweight convolution blocks connected with residuals. Each convolution block is composed of a convolution 2D layer, Group Normalization, and GeLU non-linear activation function. Then we apply a 2D point-wise convolution layer and set the output channel to 1, thus mapping the input latent feature to a 2D scalar value output map with the same spatial size.

We train our one-step text-to-image model on Stable Diffusion XL using these two choices of discriminator respectively, while keeping the other settings unchanged. Our experiments reveal that the latent-based discriminator not only reduces the training cost but also provides more visually compelling generation results. Additionally, note that the teacher diffusion model applied for phased consistency distillation is actually a pre-trained U-Net. And we freeze the parameters of U-Net for the training discriminator. Therefore, we can apply the same U-Net, which simultaneously works as the teacher diffusion model for numerical ODE solver computation and discriminator visual backbone.

VI.4.2 Video Generation

For the video discriminator, we mainly follow our design on the image discriminator. We use the same U-Net with temporal inflation as the visual backbone for video latent code as well as the teacher video diffusion model for phased consistency distillation. We still apply the 2D convolution layers as discriminator heads for each frame of hidden features extracted from the temporal inflated U-Net since we do not observe performance gain when using 3D convolutions. Note that the temporal relationships are already processed by the pre-trained temporal inflated U-Net, therefore using simple 2D convolution layers as discriminator heads is enough for supervision of the distribution of video inputs.

VII More Generation Results.

https://doi.org/10.52202/079017-2668

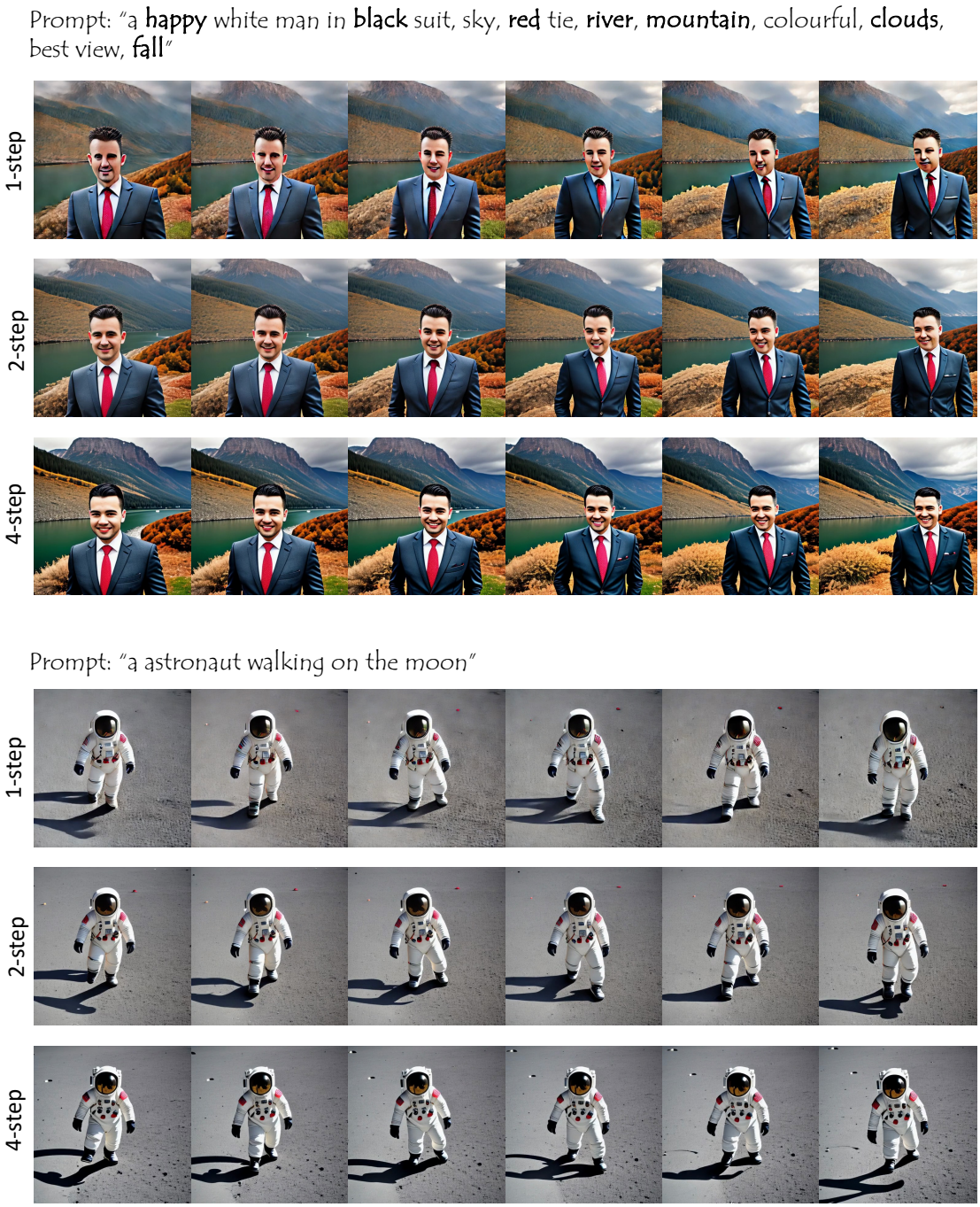

Figure 10: PCM video generation results with Stable-Diffusion v1-5 under $1 \sim 4$ inference steps.

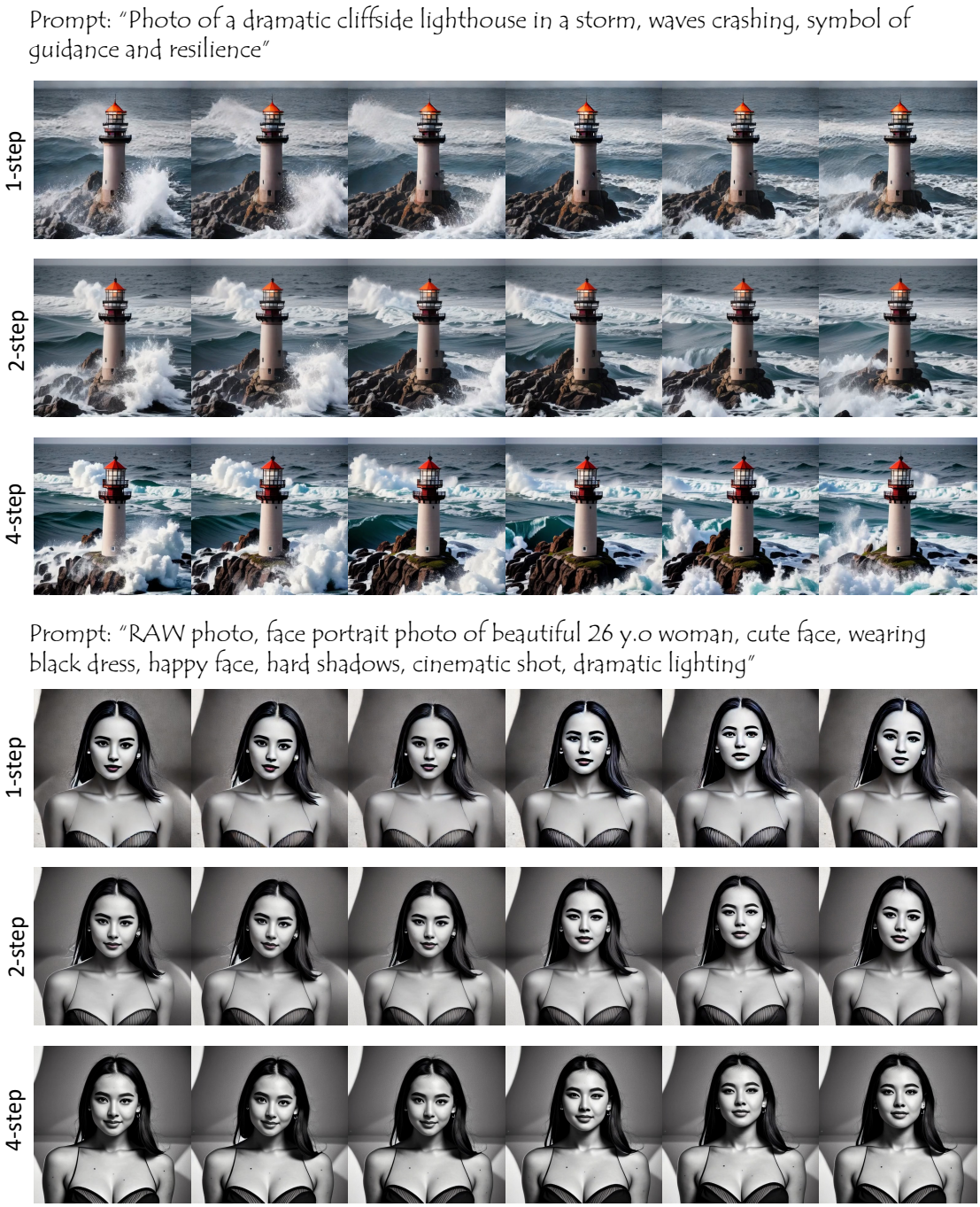

Figure 11: PCM video generation results with Stable-Diffusion v1-5 under $1 \sim 4$ inference steps.

https://doi.org/10.52202/079017-2668

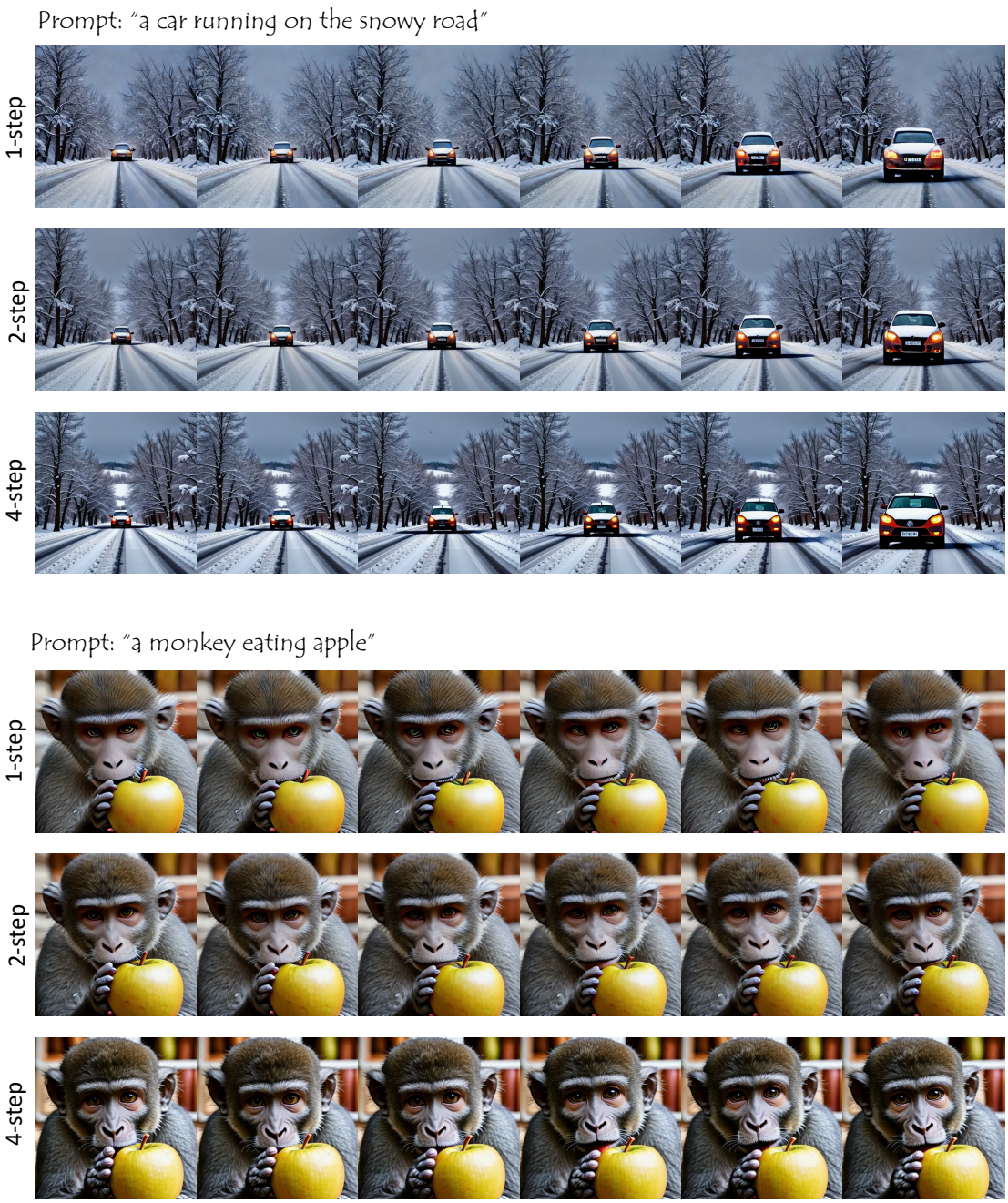

Figure 12: PCM video generation results with Stable-Diffusion v1-5 under $1 \sim 4$ inference steps.

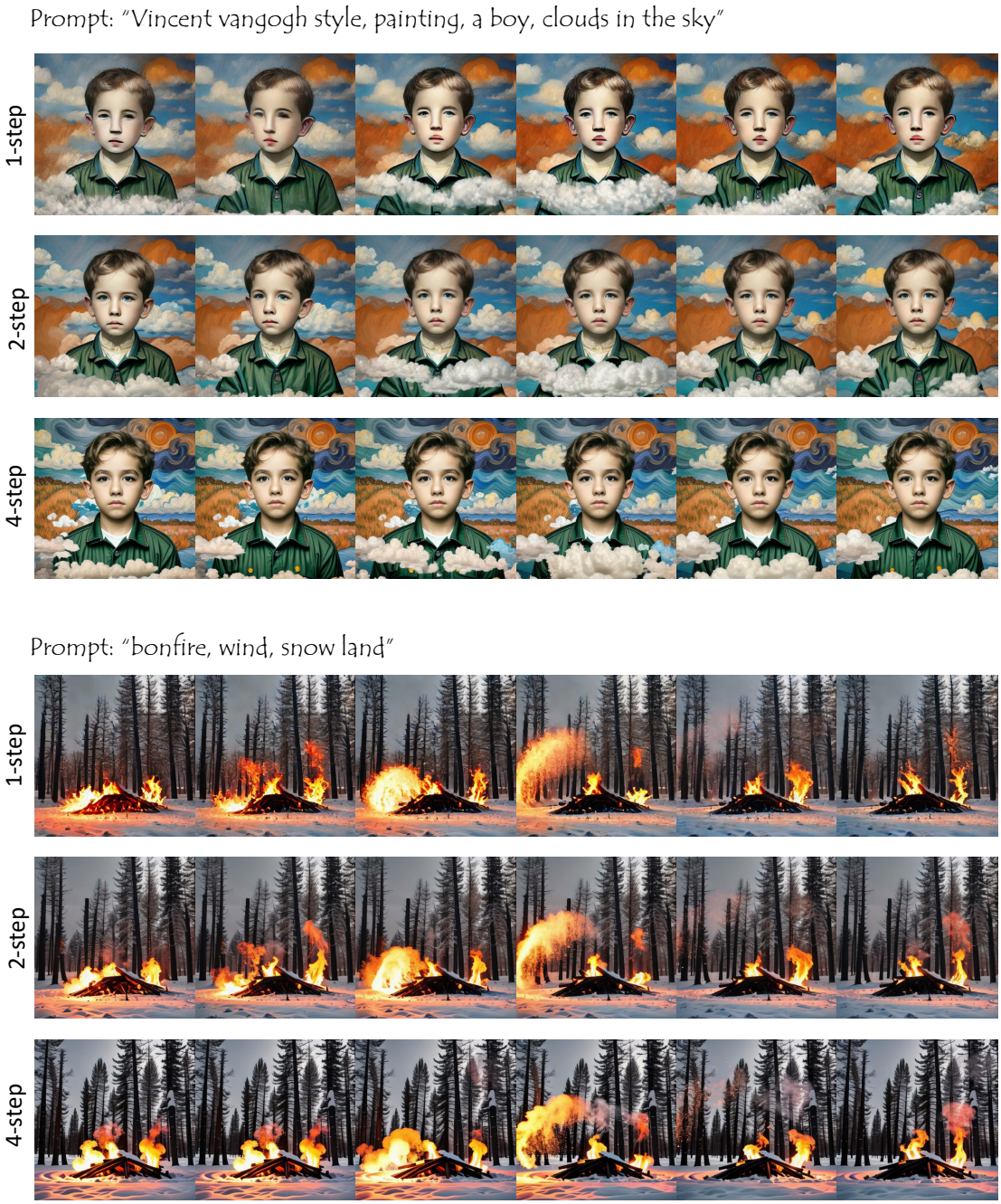

Figure 13: PCM video generation results with Stable-Diffusion v1-5 under $1 \sim 4$ inference steps.

https://doi.org/10.52202/079017-2668

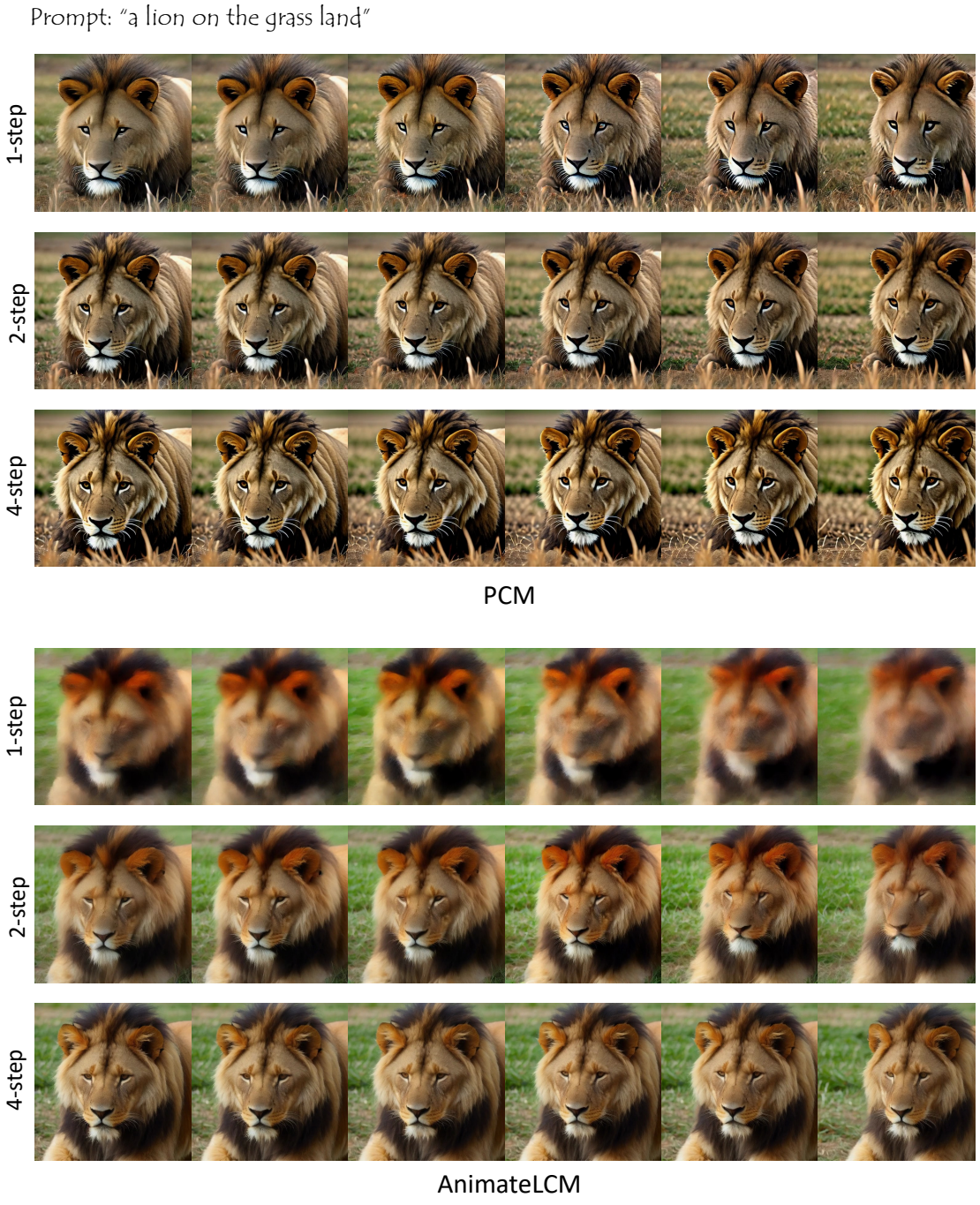

Figure 14: PCM video generation results comparison with AnimateLCM under $1 \sim 4$ inference steps.

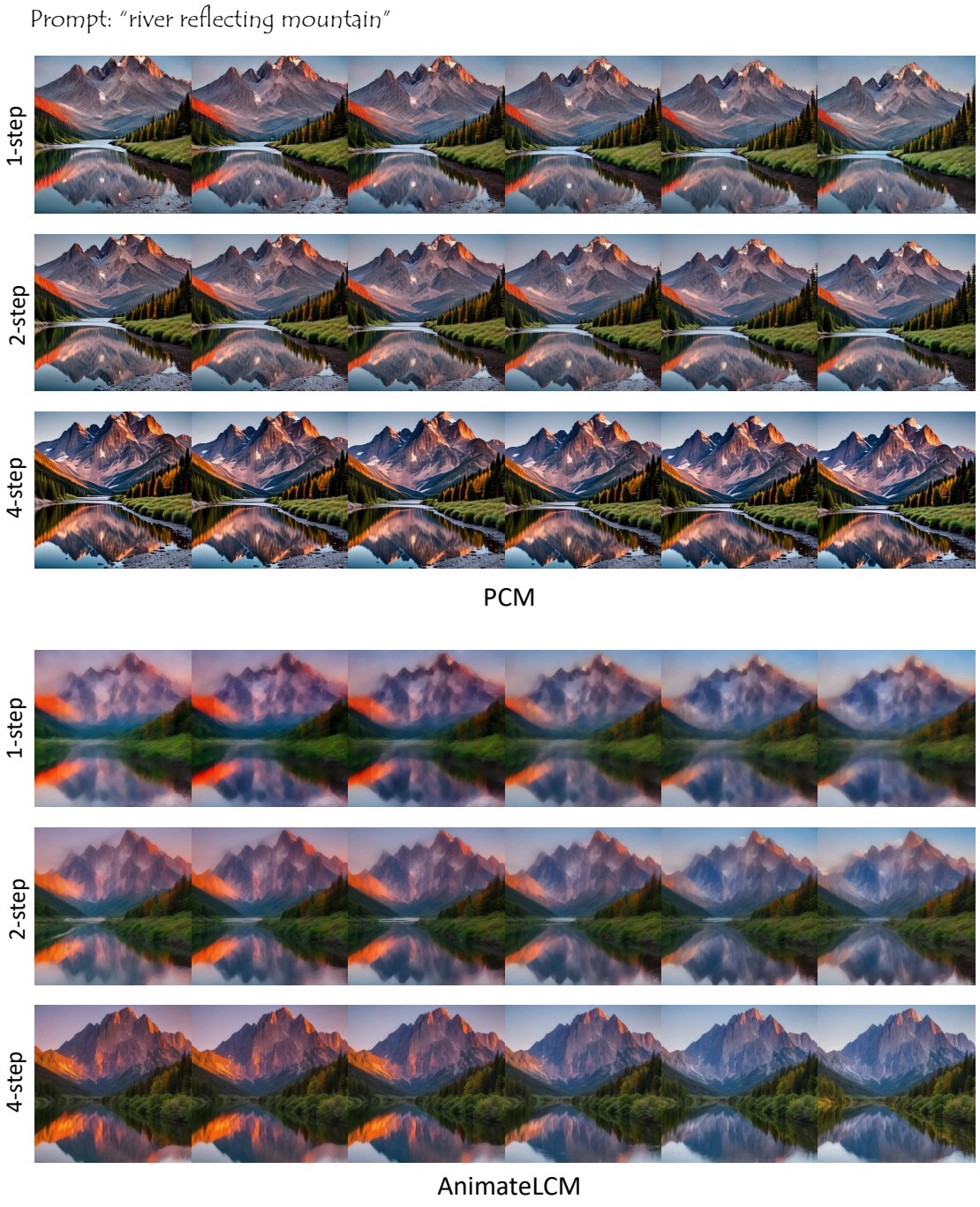

Figure 15: PCM video generation results comparison with AnimateLCM under $1 \sim 4$ inference steps.

https://doi.org/10.52202/079017-2668

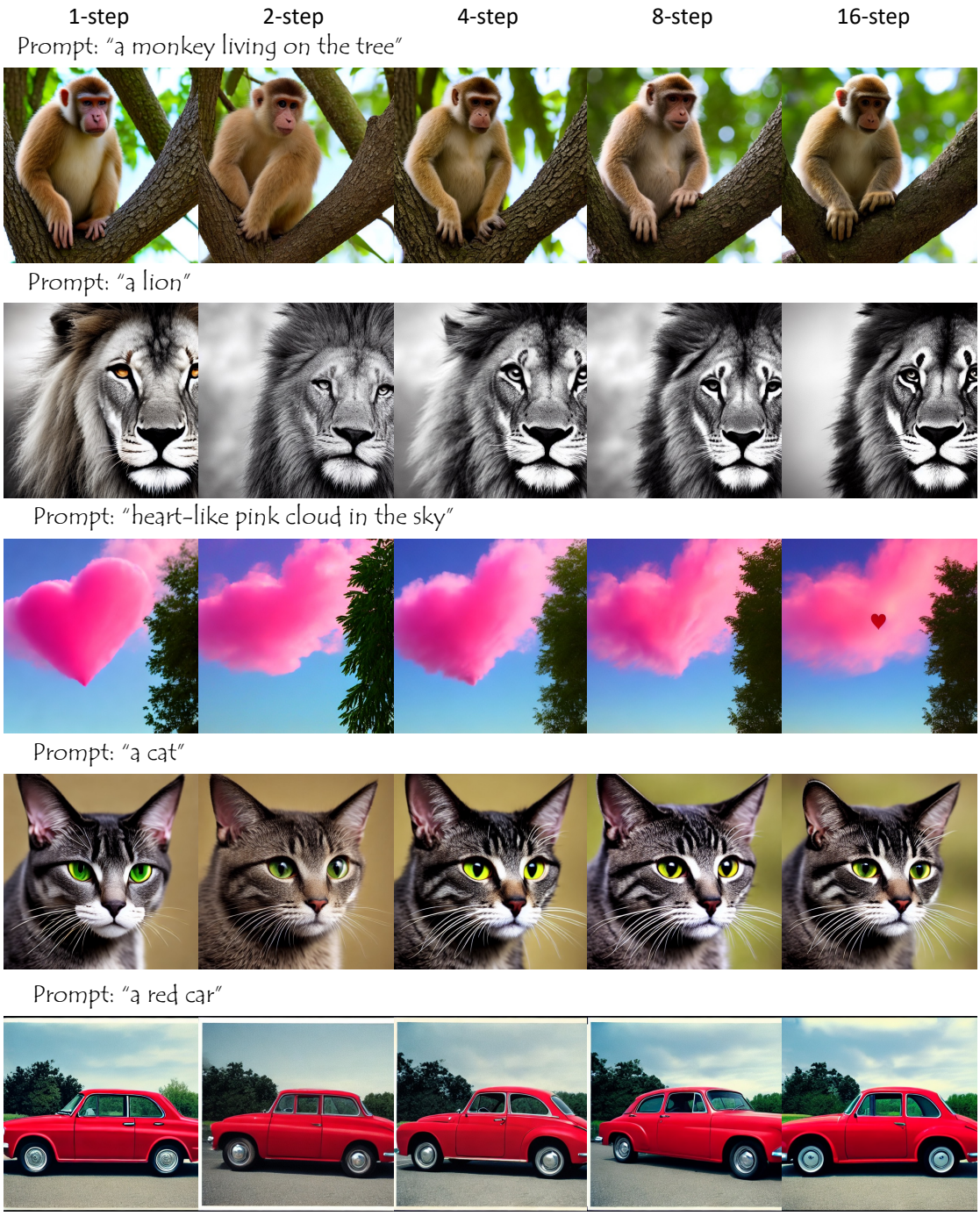

Figure 16: PCM generation results with Stable-Diffusion v1-5 under different inference steps.

https://doi.org/10.52202/079017-2668

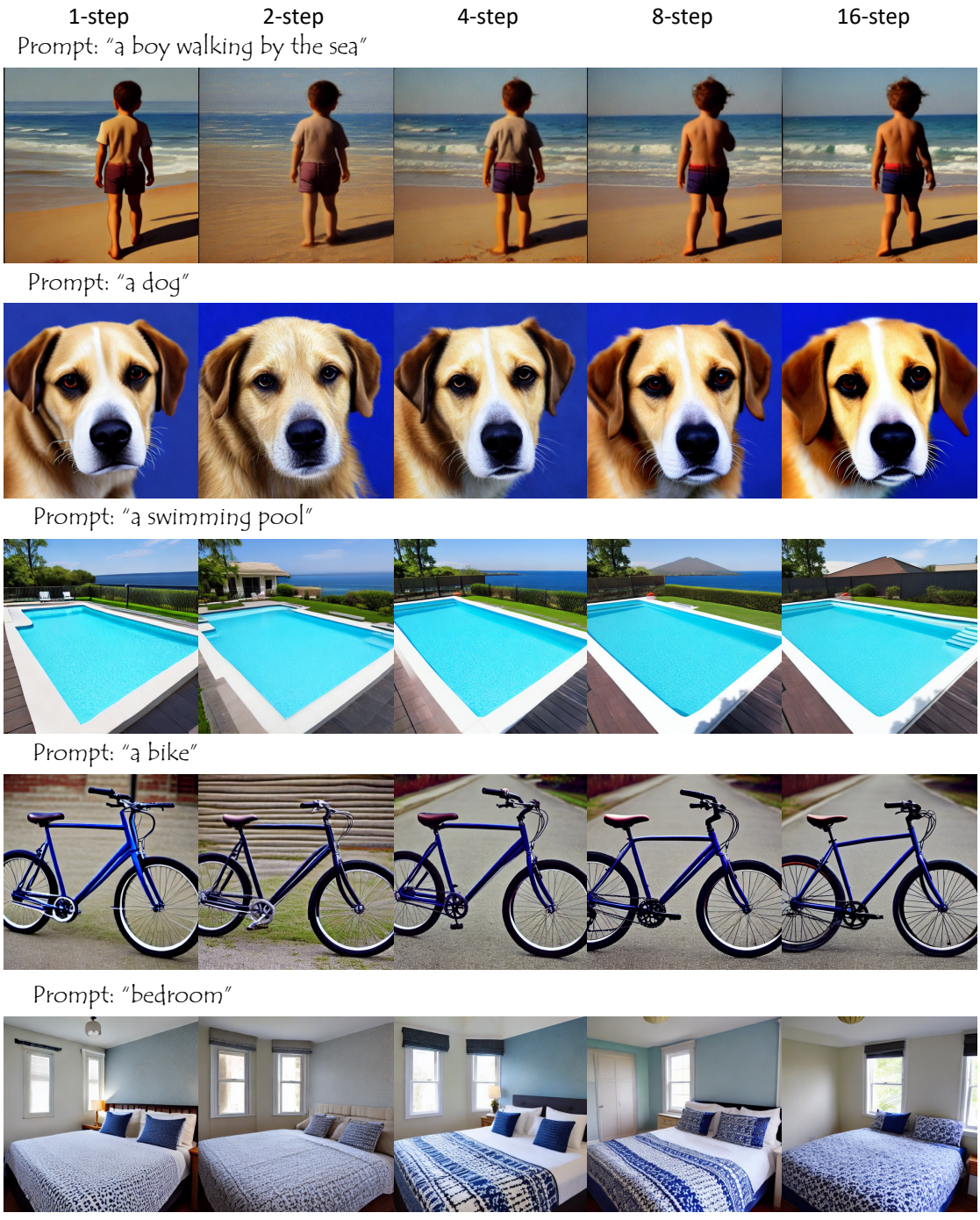

Figure 17: PCM generation results with Stable-Diffusion v1-5 under different inference steps.

https://doi.org/10.52202/079017-2668

Figure 18: PCM generation results with Stable-Diffusion XL under different inference steps.

Figure 19: PCM generation results with Stable-Diffusion XL under different inference steps.

https://doi.org/10.52202/079017-2668

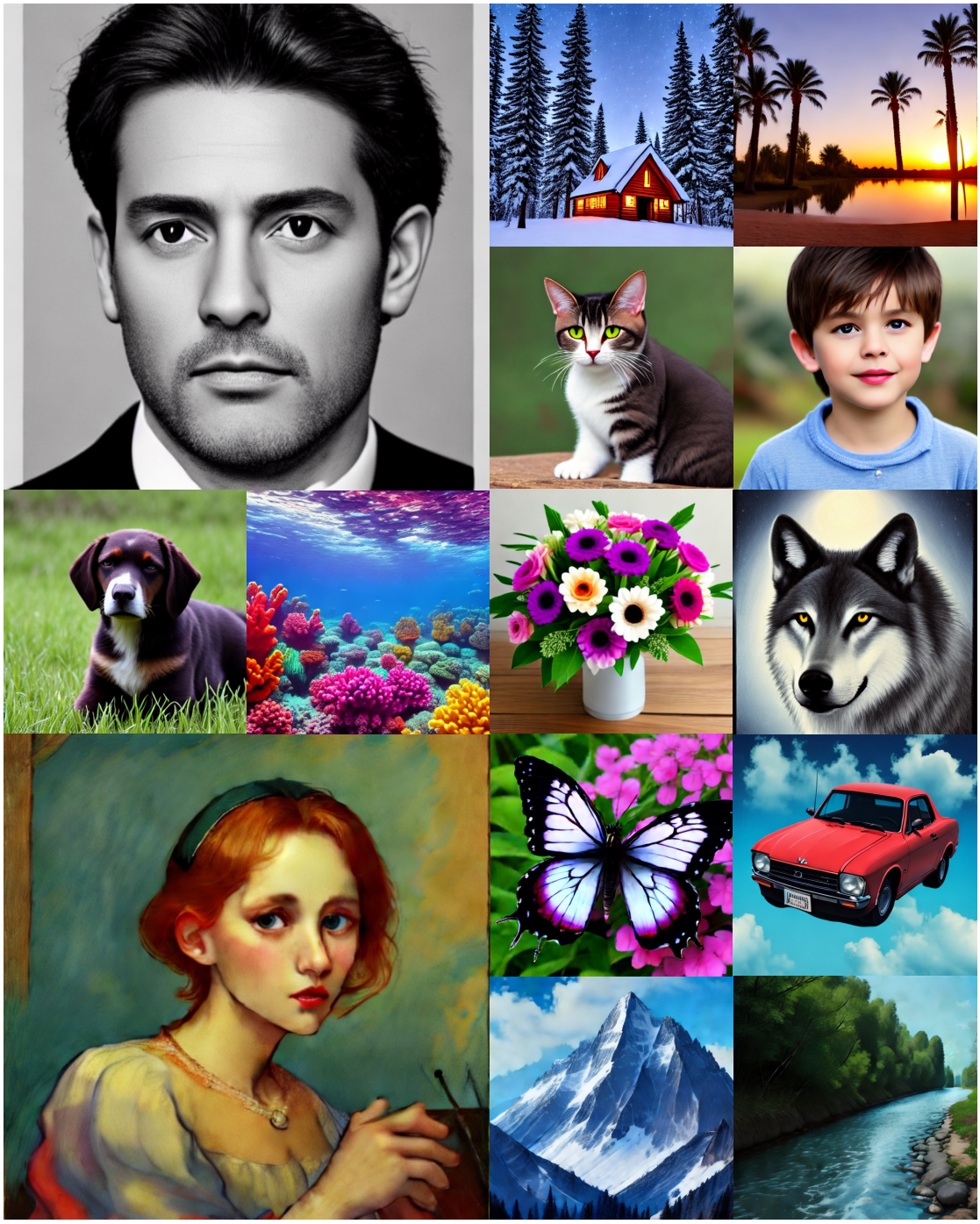

Figure 20: PCM generation 1-step results with Stable-Diffusion v1-5.

Figure 21: PCM generation 2-step results with Stable-Diffusion v1-5.

https://doi.org/10.52202/079017-2668

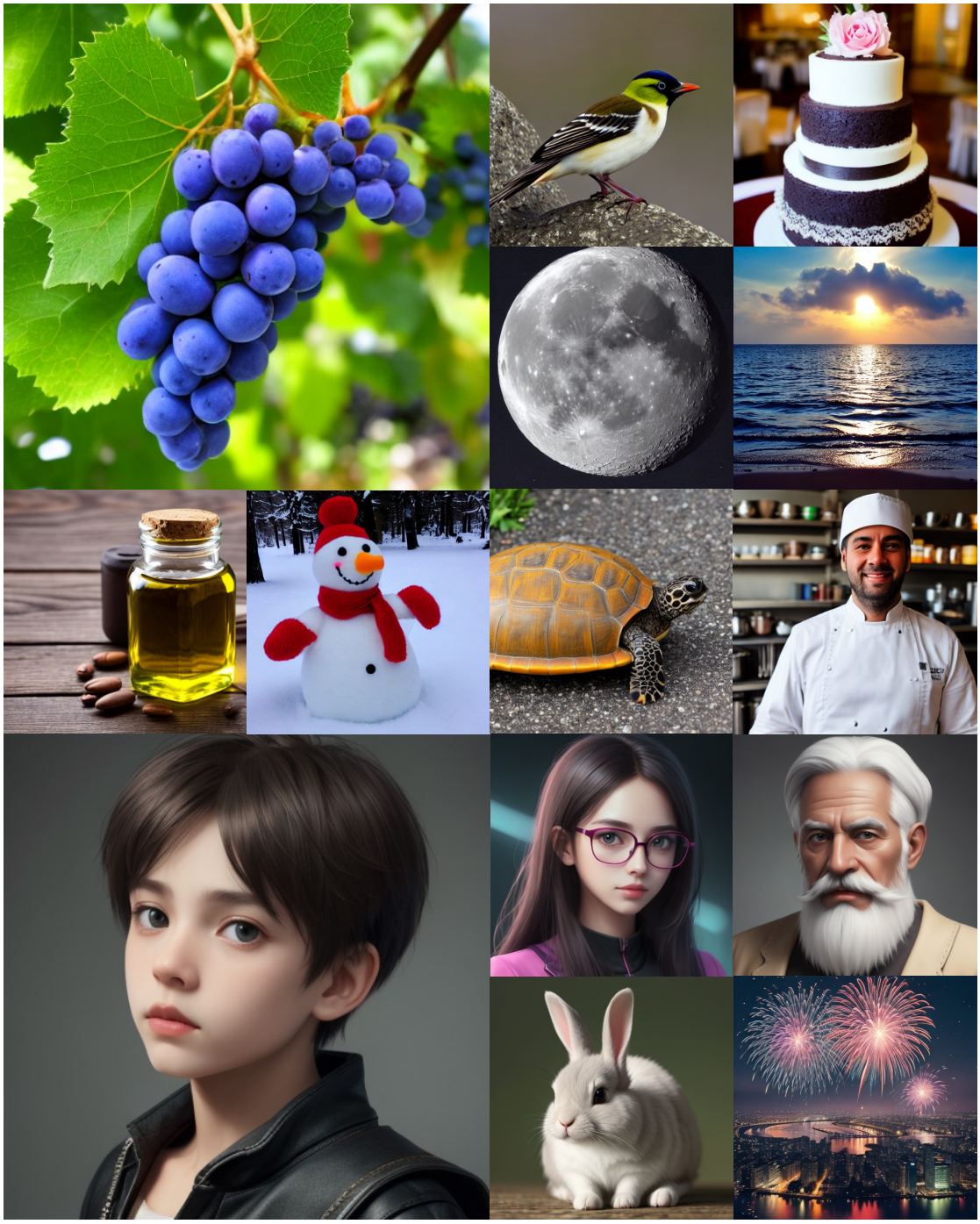

Figure 22: PCM generation 4-step results with Stable-Diffusion v1-5.

Figure 23: PCM generation 8-step results with Stable-Diffusion v1-5.

https://doi.org/10.52202/079017-2668

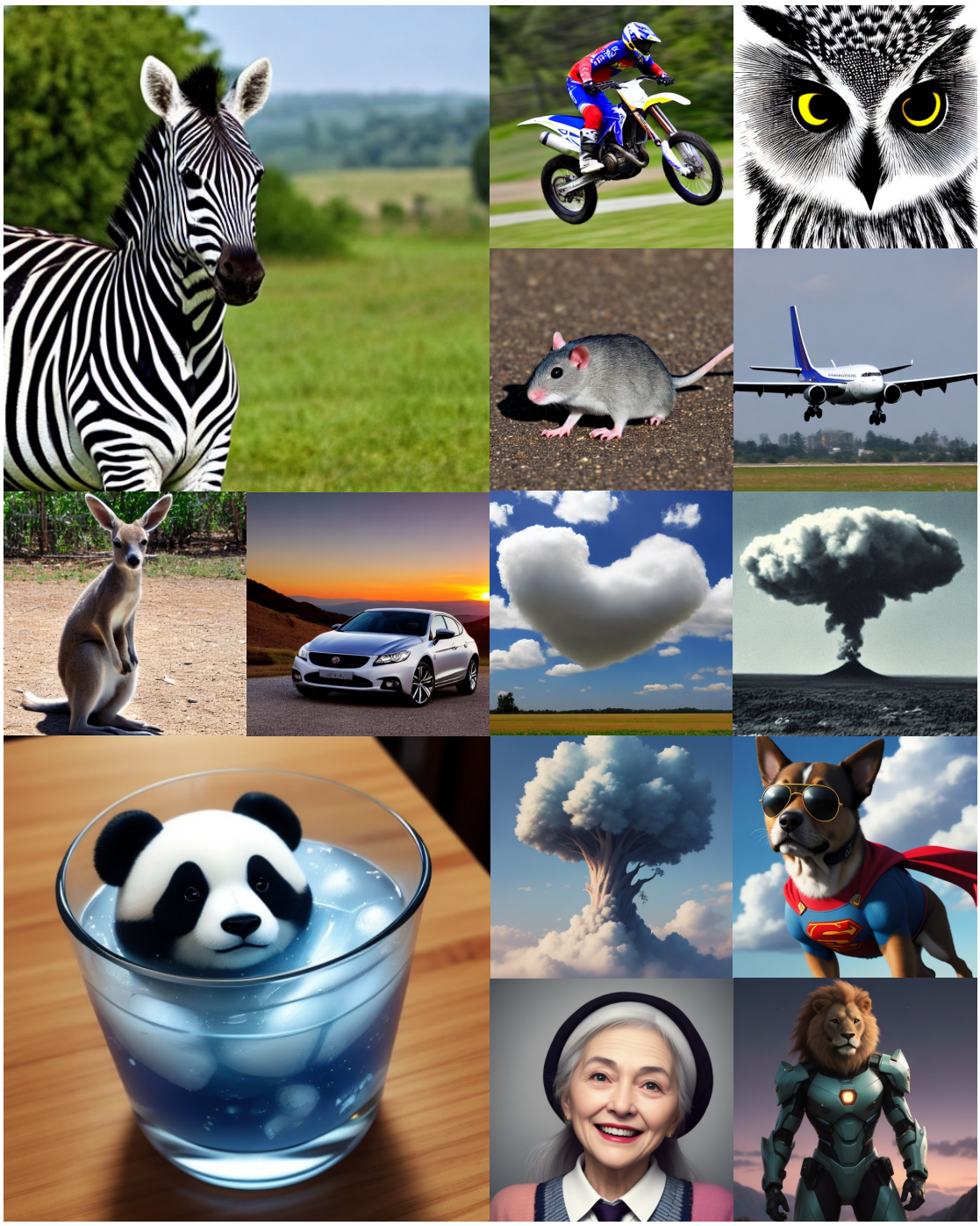

Figure 24: PCM generation 16-step results with Stable-Diffusion v1-5.

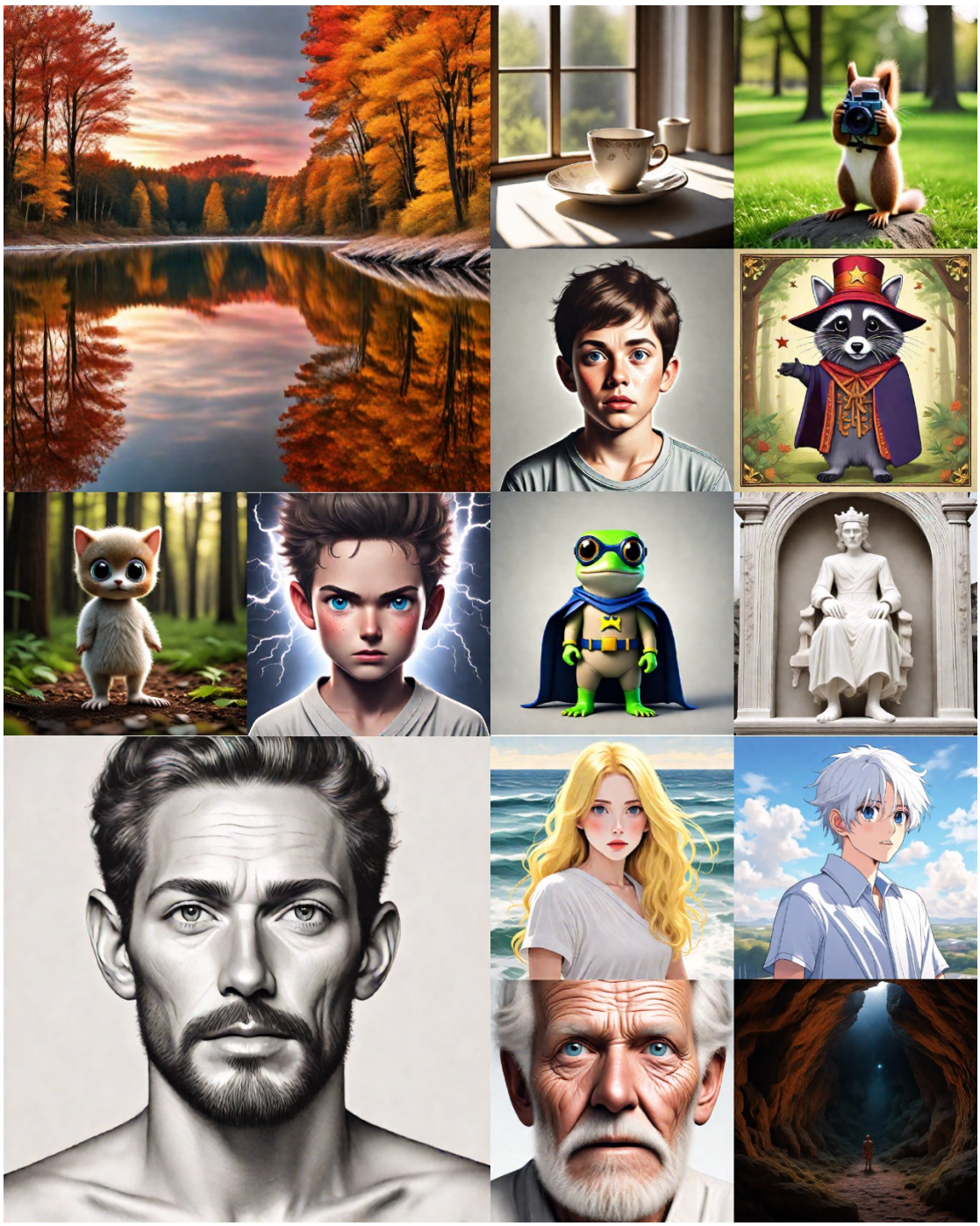

Figure 25: PCM generation 1-step results with Stable-Diffusion XL.

https://doi.org/10.52202/079017-2668

Figure 26: PCM generation 2-step results with Stable-Diffusion XL.

Figure 27: PCM generation 4-step results with Stable-Diffusion XL.

https://doi.org/10.52202/079017-2668

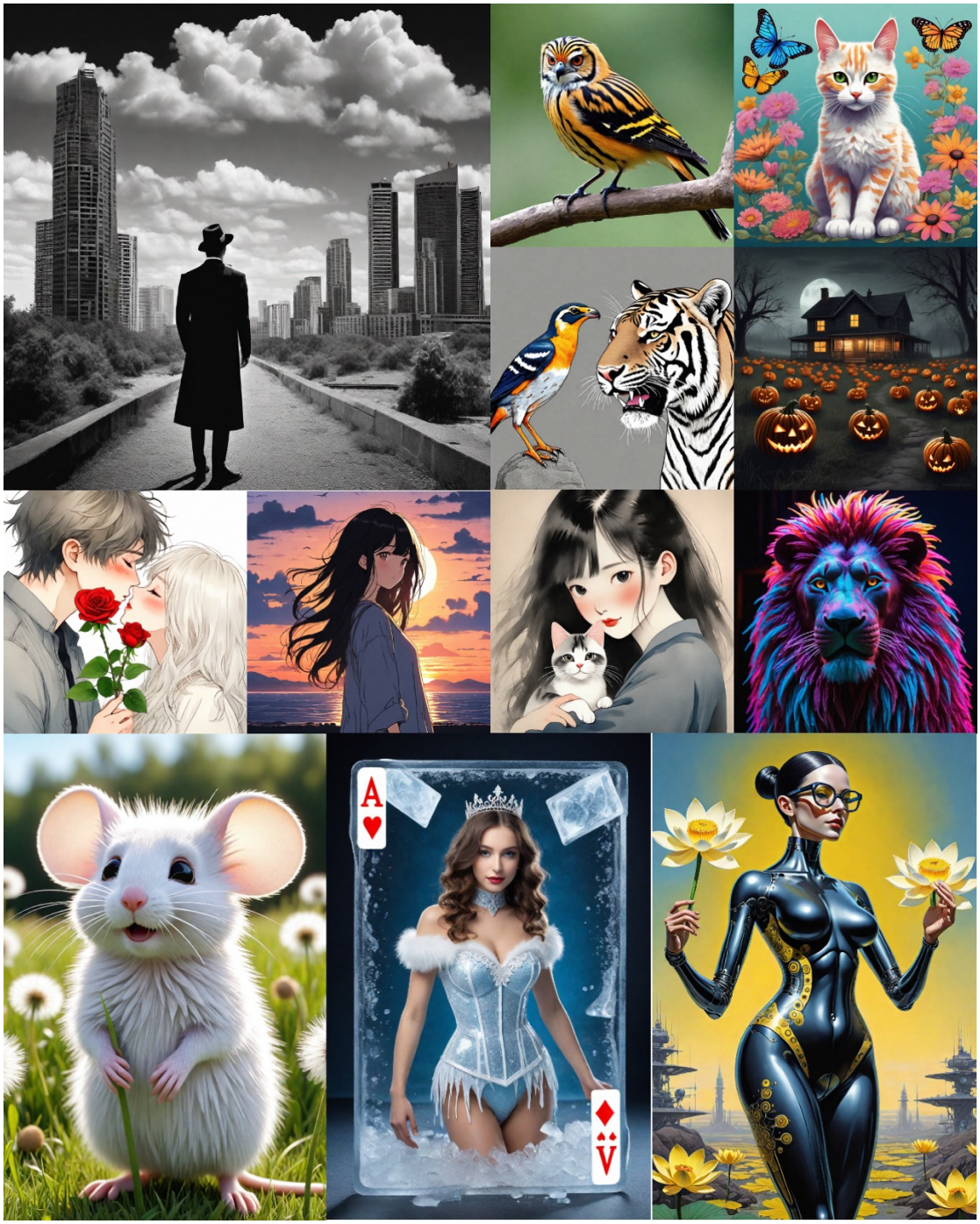

Figure 28: PCM generation 8-step results with Stable-Diffusion XL.

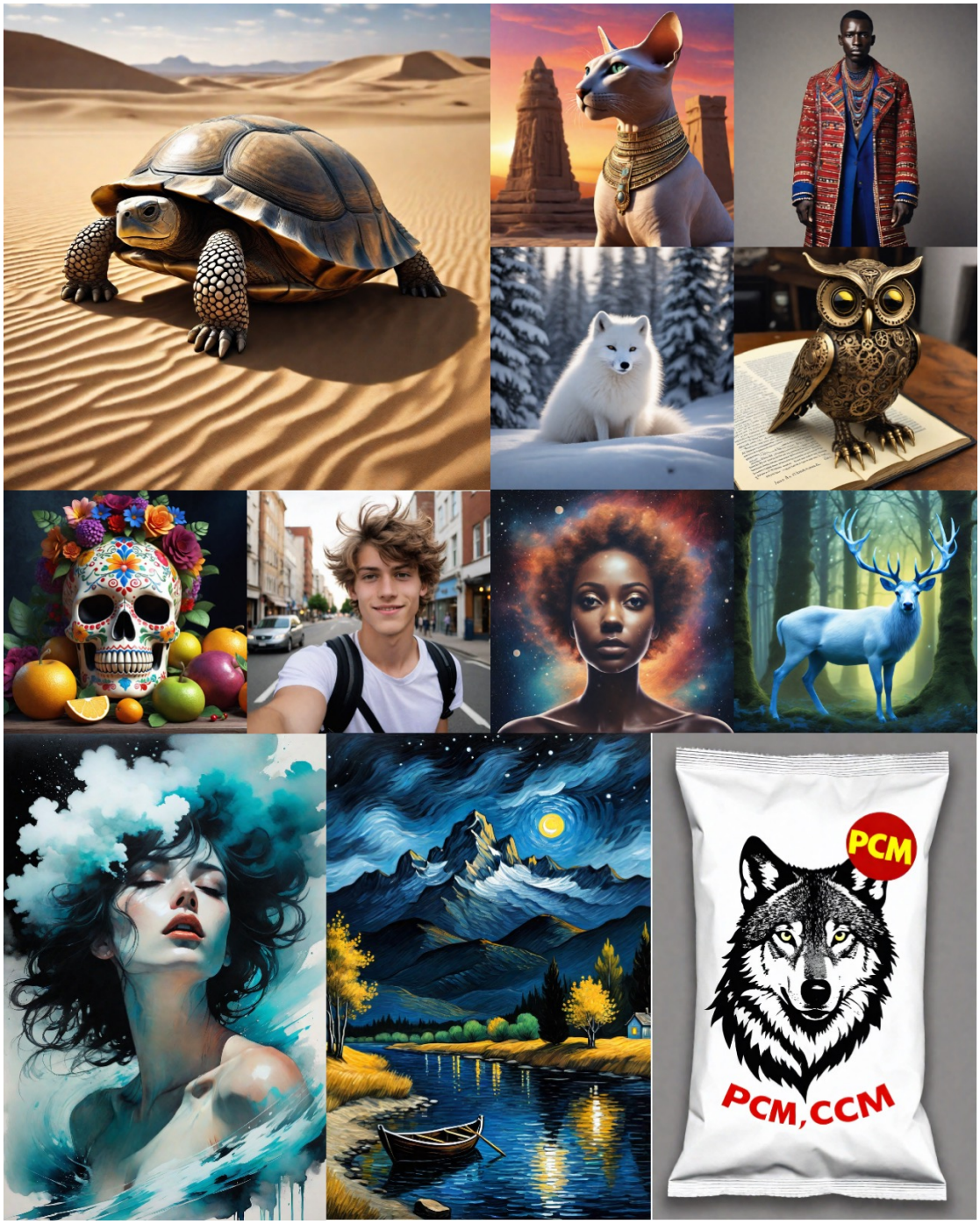

Figure 29: PCM generation 16-step results with Stable-Diffusion XL.

https://doi.org/10.52202/079017-2668

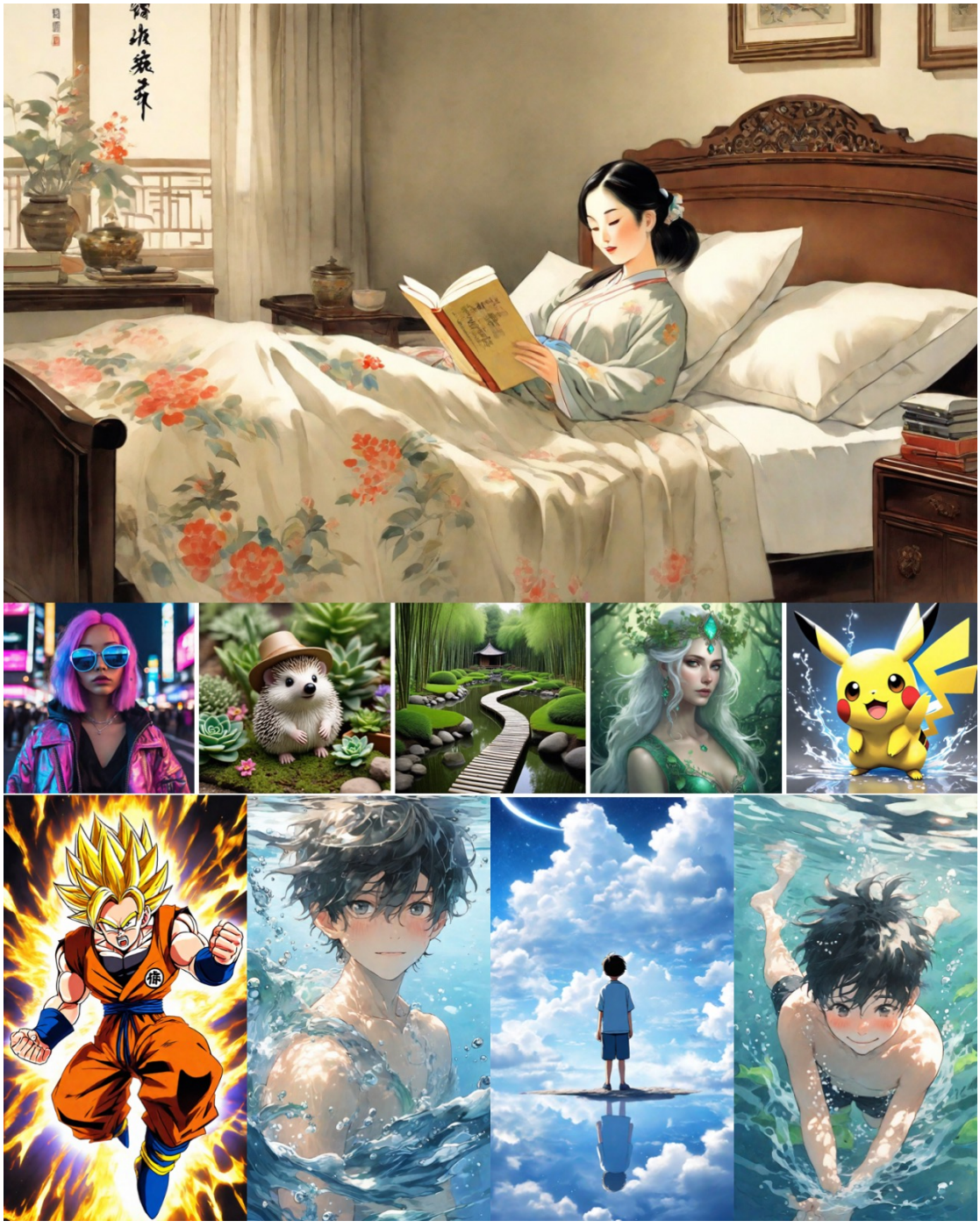

Figure 30: PCM generation 16-step results with Stable-Diffusion XL.

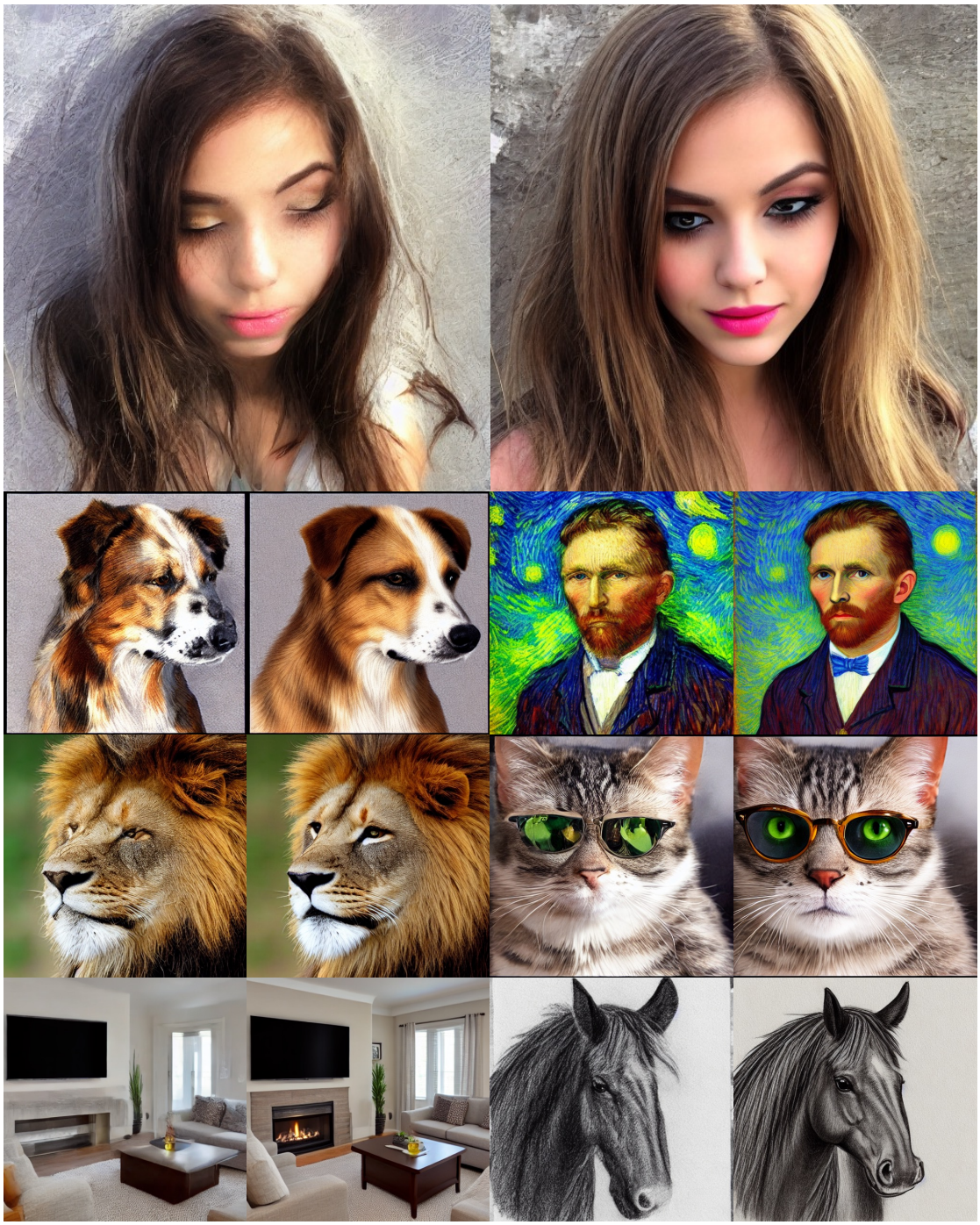

Figure 31: Visual examples of ablation study on the proposed distribution consistency loss. Left: Results generated without the distribution consistency loss. Right: Results generated with the distribution consistency loss.

https://doi.org/10.52202/079017-2668

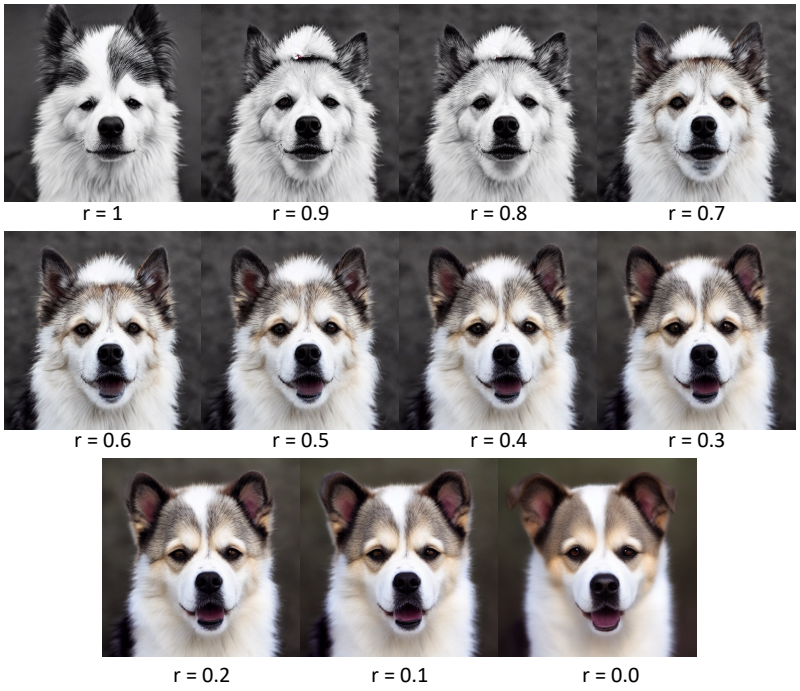

Figure 32: Visual examples of ablation study on the proposed way for stochastic sampling. Pure deterministic sampling algorithms sometimes bring artifacts in the generated results. Adding stochasticity to a certain degree can alleviate those artifacts.

https://doi.org/10.52202/079017-2668

CFG value = 1.0. The entropy of color: 1.75

CFG value = 0.6. The entropy of color: 2.16

Figure 33: Visual examples of ablation study on the proposed strategy for promoting diversity. Upper: Batch samples generated with prompt "a car" with normal CFG=1.0 value in 4-step. Lower: Batch sample generated with prompt "a car" with our proposed strategy with CFG=0.6.

https://doi.org/10.52202/079017-2668

CFG value = 1.0. The entropy of color: 1.30

CFG value = 0.6. The entropy of color: 1.75

Figure 34: Visual examples of ablation study on the proposed strategy for promoting diversity. Upper: Batch samples generated with prompt "a car" with normal CFG=1.0 value in 4-step. Lower: Batch sample generated with prompt "a car" with our proposed strategy with CFG=0.6.

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: We have claimed in the abstract and introduction that, in this paper, we identify three key flaws in the current design of LCM. We investigate the reasons behind these limitations and propose the Phased Consistency Model (PCM), which generalizes the design space and addresses all identified limitations. Accordingly, in Lines 32 to 42 in the introduction of the main paper, we reveal the limitations of LCM, and in Sec. 3 we propose PCM.

   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 work in Sec. 5

   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**

https://doi.org/10.52202/079017-2668

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

Answer: [Yes]

Justification: We present a complete proof in Sec. II.

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: We have carefully introduced the framework architecture in Sec. 3.1, the training strategy in Sec. 3.2 and Sec. 3.3. Also, we enclose the pseudo training code, the detailed framework design in the Appendix. Eventually, We will release the code and model of this paper.

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: Code is available at `https://github.com/G-U-N/Phased-Consistency-Model`.

   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: We specify all the training and test details in Sec. VI, including the training and testing details in Sec. VI.1, the pseudo training scripts in Sec. VI.2, and additional details in Sec. VI.3 and Sec. VI.4.

   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: Error bars are not reported, because it would be too computationally expensive. However, we indeed present plenty of qualitative results to demonstrate the statistical significance of the experiments.

   Guidelines:

https://doi.org/10.52202/079017-2668

- 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 have clarified in the Abstract and Sec. III that our results are achieved by only 8 A800 GPUs.

   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 conforms, 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: The positive societal impacts are discussed in Sec. V.1, and the negative societal impacts are discussed in Sec. V.2.

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, we are the image generators, which naturally have a risk of generating harmful content. Therefore, in Sec. V.3, we introduce our solution to avoid harmful content. We will include the safeguards in our pipeline when releasing the code and model.

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: All the assets used in this paper are public datasets, allowing for research use. Specifically, the used datasets include:

- CC3M – custom terms reading: The dataset may be freely used for any purpose.
- WebVid-2M – custom terms reading: Non-Commercial research for 12 months,

https://doi.org/10.52202/079017-2668

- COCO-2014 – Creative Commons Attribution 4.0 License.

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: [NA]

    Justification: We do not release new asserts in the submission. But we will release the code and model shortly. We will prepare documentation alongside the assets.

    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: This 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: We do 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.

https://doi.org/10.52202/079017-2668