SANA-Sprint: One-Step Diffusion with Continuous-Time Consistency Distillation

Junsong Chen[1*] Shuchen Xue[5*] Yuyang Zhao[1†] Jincheng Yu[1†]
Sayak Paul[4] Junyu Chen[3] Han Cai[1] Song Han[1,2] Enze Xie[1]
[1]NVIDIA [2]MIT [3]Tsinghua University [4]Huggingface [5]Independent Researcher

*Equal contribution. †Core contribution.

Abstract

*This paper presents SANA-Sprint, an efficient diffusion model for ultra-fast text-to-image (T2I) generation. SANA-Sprint is built on a pre-trained foundation model and augmented with hybrid distillation, dramatically reducing inference steps from 20 to 1-4. We introduce three key innovations: **(1)** We propose a training-free approach that transforms a pre-trained flow-matching model for continuous-time consistency distillation (sCM), eliminating costly training from scratch and achieving high training efficiency. Our hybrid distillation strategy combines sCM with latent adversarial distillation (LADD): sCM ensures alignment with the teacher model, while LADD enhances single-step generation fidelity. **(2)** SANA-Sprint is a unified step-adaptive model that achieves high-quality generation in 1-4 steps, eliminating step-specific training and improving efficiency. **(3)** We integrate ControlNet with SANA-Sprint for real-time interactive image generation, enabling instant visual feedback for user interaction. SANA-Sprint establishes a new Pareto frontier in speed-quality tradeoffs, achieving state-of-the-art performance with 7.59 FID and 0.74 GenEval in only 1 step — outperforming FLUX-schnell (7.94 FID / 0.71 GenEval) while being 10× faster (0.1s vs 1.1s on H100). It also achieves 0.1s (T2I) and 0.25s (ControlNet) latency for 1024×1024 images on H100, and 0.31s (T2I) on an RTX 4090, showcasing its exceptional efficiency and potential for AI-powered consumer applications (AIPC). Code and pre-trained models will be open-sourced.*

1. Introduction

The computational intensity of diffusion generative models [15, 53], typically requiring 50-100 iterative denoising steps, has driven significant innovation by time-step distillation to enable efficient inference. Current methodologies primarily coalesce into two dominant paradigms: (1) distribution-based distillations like GAN [12] (e.g., ADD [52], LADD [51]) and its variational score distillation (VSD) variants [37, 46, 62] leverage joint training to align single-step outputs with multi-step teacher's distributions, and (2) trajectory-based distillations like Direct Distillation [35], Progressive Distillation [39, 49], Consistency Models (CMs) [54] (e.g. LCM [36], CTM [20], MCM [13], PCM [60], sCM [34]) learn ODE solution across reduced sampling intervals. Together, these methods achieve 10-100× image generation speedup while maintaining competitive generation quality, positioning distillation as a critical pathway toward practical deployment.

Despite their promise, key limitations hinder broader adoption. GAN-based methods suffer from training instability due to oscillatory adversarial dynamics and mode collapse. GANs face challenges due to the need to map noise to natural images without supervision, making unpaired learning more ill-posed than paired learning, as highlighted in [17, 76]. This instability is compounded by architectural rigidity, which demands meticulous hyperparameter tuning when adapting to new backbones or settings. VSD-based methods involve the joint training of an additional diffusion model, which increases computational overhead and imposes significant pressure on GPU memory, and requires careful tuning [69]. Consistency models, while stable, suffer quality erosion in ultra-few-step regimes (e.g., $<$4 steps), particularly in text-to-image tasks where trajectory truncation errors degrade semantic alignment. These challenges underscore the need for a distillation framework that harmonizes efficiency, flexibility, and quality.

In this work, we present SANA-Sprint, an efficient diffusion model for one-step high-quality text-to-image (T2I) generation. Our approach builds on a pre-trained image generation model SANA and recent advancements in continuous-time consistency models (sCMs) [34], preserving the benefits of previous consistency-based models while mitigating the discretization errors of their discrete-time counterparts. To achieve the one-step generation, we first transform SANA, a Flow Matching model, to the TrigFlow model, which is required for sCM distillation, through a lossless mathematical transformation. Then, to mitigate the instability of distillation, we adapt the QK norm in self- and cross-attention in

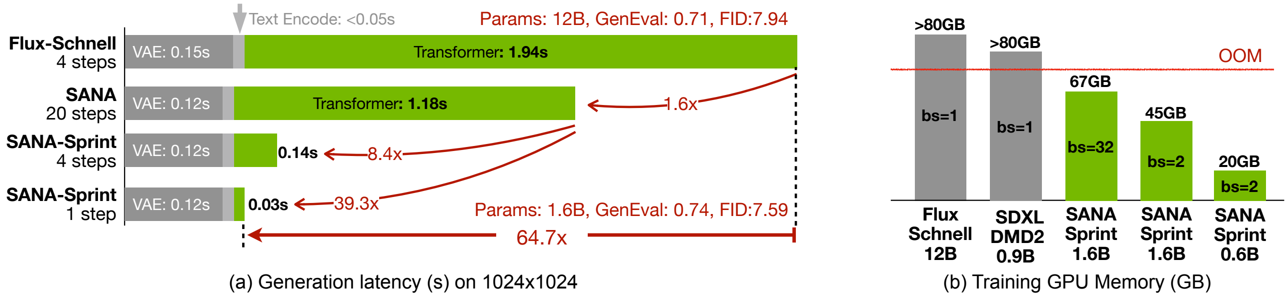


Figure 1. (a) Our SANA-Sprint accelerate the inference speed for generating 1024 $\times$ 1024 images, achieving a remarkable speedup from FULX-Schnell's 1.94 seconds to only 0.03 seconds. This represents a 64$\times$ improvement over the current state-of-the-art step-distilled model, FLUX-Schnell, as measured with a batch size of 1 on an NVIDIA A100 GPU. The ratio is calculated based on Transformer latency. (b) Additionally, our model demonstrates efficient GPU memory usage during training, outperforming other distillation methods in terms of memory cost. The GPU memory is measured using official code, 1024 $\times$ 1024 images and on a single A100 GPU.

SANA along with dense time embeddings to allow efficient knowledge transfer from the pre-trained models without retraining the teacher model. We further combine sCM with LADD's adversarial distillation to enable fast convergence and high-fidelity generation while retaining the advantages of sCMs. Note that, although validated primarily on SANA, our method can benefit other mainstream flow-matching models such as FLUX [23] and SD3 [9].

As a result, SANA-Sprint achieves excellent speed/quality tradeoff, benefiting from a hybrid objective, inheriting sCM's diversity preservation and alignment with the teacher, while integrating LADD's fidelity enhancement: experiments show a 0.6 lower FID and 0.4 higher CLIP-Score at 2-step generations compared to standalone sCM, with 3.9 lower FID and 0.9 higher CLIP-Score over pure latent adversarial approaches. As shown in Fig. 1, SANA-Sprint achieves state-of-the-art performance in FID and GenEval benchmark, surpassing recent advanced methods including SD3.5-Turbo, SDXL-DMD2, and Flux-schnell. Especially, SANA-Sprint is 64.7$\times$ faster than Flux-Schnell and exceeds in FID (7.59 vs 7.94) and GenEval (0.74 vs 0.71).

Moreover, SANA-Sprint demonstrates unprecedented inference speed—generating 1024$\times$1024 images in 0.31 seconds on a laptop with consumer-grade GPUs (NVIDIA RTX 4090) and 0.1 seconds on H100 GPU, 8.4$\times$ speedup than teacher model SANA. This efficiency unlocks transformative applications that require instant visual feedback: in ControlNet-guided image generation/editing, by integrating with ControlNet, SANA-Sprint enables instant interaction with 250ms latency on H100. SANA-Sprint exhibits robust scalability and is potentially suitable for human-in-the-loop creative workflows, AIPC, and immersive AR/VR interfaces. In summary, our key contributions are threefold:

- **Hybrid Distillation Framework**: We designed an innovative hybrid distillation framework that seamlessly transforms the flow model into the Trigflow model, integrating continuous-time consistency models (sCM) with latent adversarial diffusion distillation (LADD). This framework leverages sCM's diversity preservation and alignment with the teacher alongside LADD's fidelity enhancement, enabling unified step-adaptive sampling.
- **Excellent Speed/Quality Tradeoff**: SANA-Sprint achieves exceptional performance with only 1-4 steps. SANA-Sprint generates a 1024×1024 image in only 0.10s-0.18s on H100, achieving state-of-the-art 7.59 FID on MJHQ-30K and 0.74 GenEval score - surpassing FLUX-schnell (7.94 FID/0.71 GenEval) while being 10$\times$ faster.
- **Real-Time Interactive Generation**: By integrating ControlNet with SANA-Sprint, we enable real-time interactive image generation in 0.25s on H100. This facilitates immediate visual feedback in human-in-the-loop creative workflows, enabling better human-computer interaction.

2. Preliminaries

2.1. Diffusion Model and Its Variants

Diffusion models [15, 53] diffuse clean data sample $\boldsymbol{x}_0 \sim p_{data}$ from data distribution to noisy data $\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_0 + \sigma_t \boldsymbol{z}$, where $t \in [0, T]$ represents time within the interval, $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is a standard Gaussian noise. The terminal distribution p_T of $\boldsymbol{x}_T$ exactly or approximately follows a Gaussian distribution. Typically, diffusion models train a noise prediction network $\boldsymbol{\epsilon_\theta}$ using $\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{z},t}[\|\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_t, t) - \boldsymbol{z}\|^2]$, which is equivalent to denoising score matching loss [53, 59]. The sampling process of diffusion models involves solving the probability flow ODE (PF-ODE) [53] $\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\boldsymbol{x}_t + (\frac{\mathrm{d}\sigma_t}{\mathrm{d}t} - \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t)\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_t, t)$ with the initial value $\boldsymbol{x}_1 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Below, we will introduce two recent formulations of diffusion models that have received significant attention.

Flow Matching [29, 31, 43] considers a linear interpolation noising process by defining $\alpha_t = 1 - t, \sigma_t = t, T = 1$. The flow matching models train a velocity prediction network $\boldsymbol{v_\theta}$ using $\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{z},t}[w(t)\|\boldsymbol{v_\theta}(\boldsymbol{x}_t, t) - (\boldsymbol{z} - \boldsymbol{x}_0)\|^2]$, where $w(t)$ is a weighting function. The sampling of flow models solves the PF-ODE $\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \boldsymbol{v_\theta}(\boldsymbol{x}_t, t)$ with the initial value

$\boldsymbol{x}_1 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.

TrigFlow [1, 34] considers a spherical linear interpolation noising process by defining $\alpha_t = \cos(t), \sigma_t = \sin(t), T = \frac{\pi}{2}$. Moreover, Trigflow assumes the noise $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \sigma_d^2 \boldsymbol{I})$, where σ_d represents the standard deviation of data distribution p_{data}. The TrigFlow models train a velocity prediction network $\boldsymbol{F_\theta}$ using $\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{z},t}[w(t)\|\sigma_d \boldsymbol{F_\theta}(\frac{\boldsymbol{x}_t}{\sigma_d}, t) - (\cos(t)\boldsymbol{z} - \sin(t)\boldsymbol{x}_0)\|^2]$, where $w(t)$ is a weighting function. The sampling of TrigFlow models solves the PF-ODE defined by $\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \sigma_d \boldsymbol{F_\theta}(\frac{\boldsymbol{x}_t}{\sigma_d}, t)$ starting from $\boldsymbol{x}_{\frac{\pi}{2}} \sim \mathcal{N}(\mathbf{0}, \sigma_d^2 \boldsymbol{I})$.

Diffusion, Flow Matching, and TrigFlow are all continuous-time generative models that differ in their interpolation schemes and velocity field parameterizations.

2.2. Consistency Models

A consistency model (CM) [54] parameterizes a neural network $\boldsymbol{f_\theta}(\boldsymbol{x}_t, t)$ which is trained to predict the solution $\boldsymbol{x}_0$ of the PF-ODE, which is the terminal clean data along the trajectory of the PF-ODE (regardless of its position in the trajectory), starting from the noisy observation $\boldsymbol{x}_t$. The conventional approach parameterizes the CM using skip connections, bearing a close resemblance to [2, 18]

$$\boldsymbol{f_\theta}(\boldsymbol{x}_t, t) = c_{\text{skip}}(t)\boldsymbol{x}_t + c_{\text{out}}(t)\boldsymbol{F_\theta}(\boldsymbol{x}_t, t), \tag{1}$$

where $c_{\text{skip}}(t)$ and $c_{\text{out}}(t)$ are differentiable functions satisfying $c_{\text{skip}}(0) = 1$ and $c_{\text{out}}(0) = 0$ to ensure the boundary conditions $\boldsymbol{f_\theta}(\boldsymbol{x}_0, 0) = \boldsymbol{x}_0$. $\boldsymbol{F_\theta}$ indicates the pretrained diffusion/flow model and $\boldsymbol{f_\theta}$ is the data prediction model. Based on the training approach, CMs can be categorized into two types: discrete-time [36, 54] and continuous-time [34, 54]. Discrete-time CMs are trained with the following objective

$$l_{CM}^{\Delta t} = \mathbb{E}_{\boldsymbol{x}_t,t}[d(\boldsymbol{f_\theta}(\boldsymbol{x}_t, t), \boldsymbol{f_{\theta^-}}(\boldsymbol{x}_{t-\Delta t}, t - \Delta t))], \tag{2}$$

where $\boldsymbol{\theta}^-$ is the `stopgrad` version of $\boldsymbol{\theta}$, $w(t)$ is the weighting function, Δt is a small time interval, and $\boldsymbol{x}_{t-\Delta t}$ is obtained from $\boldsymbol{x}_t$ by running a numerical ODE solver. $d(\cdot,\cdot)$ is a metric such as ℓ_1, squared ℓ_2, Pseudo-Huber loss, and the LPIPS loss [73].

Although discrete-time CMs work well in practice, the additional discretization errors brought by numerical ODE solvers are inevitable. Continuous-time CMs correspond to the limiting case of $\Delta t \to 0$ in Eq. (2). When choosing $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2^2$, the expression simplifies to:

$$l_{CM}^{cont.} := \lim_{\Delta t \to 0} \frac{l_{CM}^{\Delta t}}{\Delta t} = \mathbb{E}_{\boldsymbol{x}_t,t}\left[w(t)\langle \boldsymbol{f_\theta}(\boldsymbol{x}_t, t), \frac{\mathrm{d}\boldsymbol{f_{\theta^-}}}{\mathrm{d}t}(\boldsymbol{x}_t, t)\rangle\right], \tag{3}$$

where $\frac{\mathrm{d}\boldsymbol{f_{\theta^-}}(\boldsymbol{x}_t,t)}{\mathrm{d}t} = \frac{\partial \boldsymbol{f_{\theta^-}}(\boldsymbol{x}_t,t)}{\partial t} + \nabla_{\boldsymbol{x}_t}\boldsymbol{f_{\theta^-}}(\boldsymbol{x}_t, t)\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t}$. The infinitesimal step of $\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t}$ replaces numerical ODE solvers, thereby eliminating discretization errors.

Specifically, under TrigFlow where $c_{\text{skip}}(t) = \cos(t)$ and $c_{\text{out}}(t) = -\sin(t)$, sCM's parameterization and arithmetic coefficients are simplified to the following form:

$$\boldsymbol{f_\theta}(\boldsymbol{x}_t, t) = \cos(t)\boldsymbol{x}_t - \sin(t)\sigma_d \boldsymbol{F_\theta}(\frac{\boldsymbol{x}_t}{\sigma_d}, t), \tag{4}$$

and the time derivative becomes:

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{f_{\theta^-}}(\boldsymbol{x}_t, t)}{\mathrm{d}t} = &- \cos(t)\left(\sigma_d \boldsymbol{F_{\theta^-}}(\frac{\boldsymbol{x}_t}{\sigma_d}, t) - \frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t}\right) \\ &- \sin(t)\left(\boldsymbol{x}_t + \sigma_d \frac{\mathrm{d}\boldsymbol{F_{\theta^-}}(\frac{\boldsymbol{x}_t}{\sigma_d}, t)}{\mathrm{d}t}\right). \end{aligned} \tag{5}$$

3. Method

sCM [34] simplify continuous-time CMs using the TrigFlow formulation. While this provides an elegant framework, most score-based generative models are based on diffusion or flow matching formulations. One possible approach is to develop separate training algorithms for continuous-time CMs under these formulations, but this requires distinct algorithm designs and hyperparameter tuning, increasing complexity. Alternatively, one could pretrain a dedicated TrigFlow model, as in [34], but this significantly increases computational cost.

To address these challenges, we propose a simple method to transform a pre-trained flow matching model into a TrigFlow model through straightforward mathematical input and output transformations. This approach makes it possible to strictly follow the training algorithm in [34], eliminating the need for separate algorithm designs while fully leveraging existing pre-trained models. The transformation process for general diffusion models can be carried out in a similar manner, which we omit here for simplicity.

3.1. Training-Free Transformation to TrigFlow

Score-based generative models (diffusion, flow matching, and TrigFlow) can denoise data with proper data scales and signal-to-noise ratios (SNRs)[1] aligned with training. However, flow matching cannot directly denoise TrigFlow-scheduled data due to three mismatches: First, their time domains differ: TrigFlow uses $[0, \frac{\pi}{2}]$, while flow matching is defined on $[0, 1]$. Second, their noise schedules are distinct—TrigFlow maintains $\cos^2(t_{\texttt{Trig}}) + \sin^2(t_{\texttt{Trig}}) = 1$, while flow matching yields $t_{\texttt{FM}}^2 + (1 - t_{\texttt{FM}})^2 < 1$, causing data scale discrepancies. Finally, their prediction targets differ: flow matching predicts $z - x_0$ with static coefficients $(1, -1)$, whereas TrigFlow predicts $\cos(t)z - \sin(t)x_0$ with time-varying coefficients. These mismatches in temporal parameterization, SNR, and output necessitate explicit input/output transformations.

To clarify, we use the subscript $_{\texttt{Trig}}$ to denote noisy data under the TrigFlow framework and $_{\texttt{FM}}$ to denote noisy data

[1] For a diffusion model $\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_0 + \sigma_t \boldsymbol{z}$, SNR is defined as $\frac{\alpha_t^2}{\sigma_t^2}$

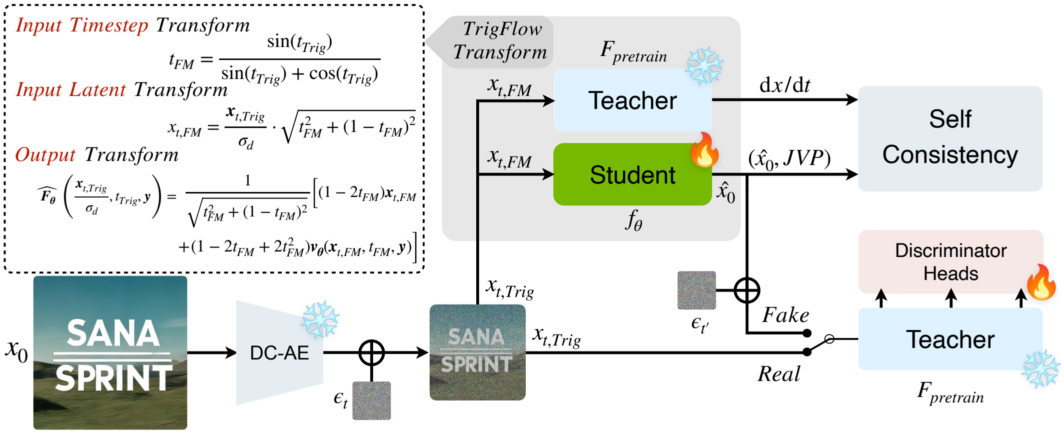


Figure 2. **Training paradigm of SANA-Sprint.** In SANA-Sprint, we use the student model for synthetic data generation ($\hat{x_0}$) and JVP calculation, and we use the teacher model for velocity ($\mathrm{d}x/\mathrm{d}t$) compute and its feature for the GAN loss, which allows us train sCM and GAN together and have only one training model purely in the latent space. Details of training objective and *TrigFlow Transformation* are in Eq. (9), Eq. (11) and Sec. 3.1.

under the flow matching framework. The following proposition outlines the transformation from flow matching models to TrigFlow models, which is theoretically lossless.

Remark. *We prioritize seamlessly transforming existing noise schedules,* e.g. *flow matching, into TrigFlow while integrating the sCM framework with minimal modifications. This approach avoids the need for pre-training a dedicated TrigFlow model, as in [34], although it involves a deviation from the unit variance principle in [18, 34].*

Proposition 3.1. *Given a noisy data $\frac{\boldsymbol{x}_{t,\texttt{Trig}}}{\sigma_d}$ under TrigFlow noise schedule, a flow matching model can denoise it via $\boldsymbol{v_\theta}(\boldsymbol{x}_{t,\texttt{FM}}, t_{\texttt{FM}}, \boldsymbol{y})$, where*

$$t_{\texttt{FM}} = \frac{\sin(t_{\texttt{Trig}})}{\sin(t_{\texttt{Trig}}) + \cos(t_{\texttt{Trig}})}, \tag{6}$$

$$x_{t,\texttt{FM}} = \frac{\boldsymbol{x}_{t,\texttt{Trig}}}{\sigma_d} \cdot \sqrt{t_{\texttt{FM}}^2 + (1-t_{\texttt{FM}})^2}. \tag{7}$$

Given $\boldsymbol{v_\theta}(\boldsymbol{x}_{t,\texttt{FM}}, t_{\texttt{FM}}, \boldsymbol{y})$, the best estimator for the TrigFlow model $\boldsymbol{F_\theta}$ is the following:

$$\begin{aligned}\widehat{\boldsymbol{F_\theta}}&\left(\frac{\boldsymbol{x}_{t,\texttt{Trig}}}{\sigma_d}, t_{\texttt{Trig}}, \boldsymbol{y}\right)\\ =&\frac{1}{\sqrt{t_{\texttt{FM}}^2 + (1-t_{\texttt{FM}})^2}}\Big[(1-2t_{\texttt{FM}})\boldsymbol{x}_{t,\texttt{FM}}\\ &+(1-2t_{\texttt{FM}}+2t_{\texttt{FM}}^2)\boldsymbol{v_\theta}(\boldsymbol{x}_{t,\texttt{FM}}, t_{\texttt{FM}}, \boldsymbol{y})\Big].\end{aligned} \tag{8}$$

Furthermore, the transformation is lossless in theory.

The details and proof of Proposition 3.1 are in Appendix D. The transformations of both input and output are all differentiable making it compatible with auto differentiation. As validated by Tab. 1, the transformation is lossless in both theory and practice. The training-free transformation is depicted in the gray box of Fig. 2.

Table 1. **Comparison of original Flow-based SANA model and training-free transformation of TrigFlow-based SANA-Sprint model.** We evaluate the FID and CLIP-Score before and after the transformation in Sec. 3.1.

Method	FID ↓	CLIP-Score ↑
Flow Euler 50 steps	5.81	28.810
TrigFlow Euler 50 steps	5.73	28.806

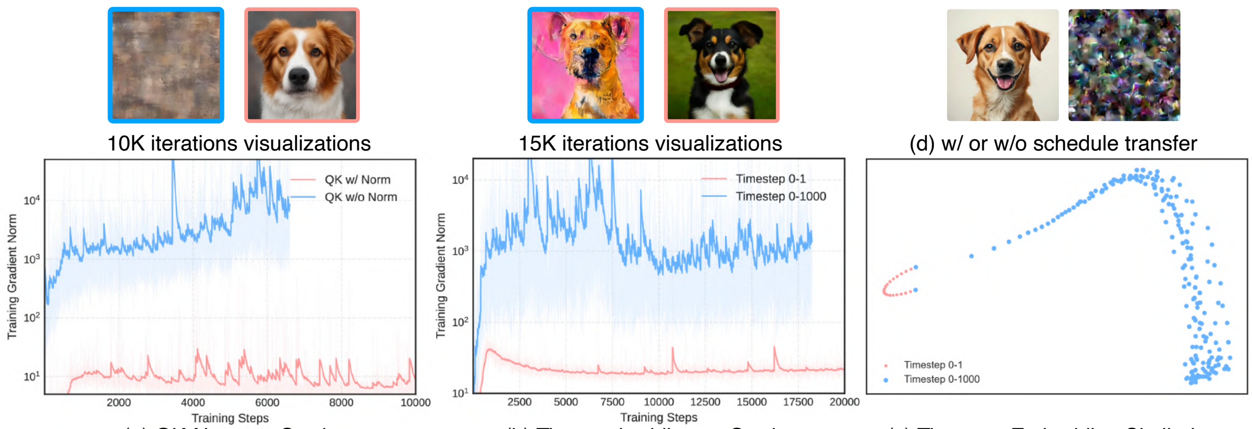
Self Consistency Loss. With the lossless transformation established, we can seamlessly adopt the training algorithm and pipeline of sCM without other modification. This allows us to directly follow the sCM training framework. Our final sCM loss is the following:

$$\begin{aligned}\mathcal{L}_{\text{sCM}}(\boldsymbol{\theta},\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{x}_t,t}\Bigg[\frac{e^{w_{\boldsymbol{\phi}}(t)}}{D}\Bigg\|\widehat{\boldsymbol{F_\theta}}\left(\frac{\boldsymbol{x}_t}{\sigma_d}, t, \boldsymbol{y}\right)\\ -\widehat{\boldsymbol{F_{\theta^-}}}\left(\frac{\boldsymbol{x}_t}{\sigma_d}, t, \boldsymbol{y}\right) - \cos(t)\frac{\mathrm{d}\widehat{\boldsymbol{f_{\theta^-}}}(\boldsymbol{x}_t, t, y)}{\mathrm{d}t}\Bigg\|_2^2 - w_{\boldsymbol{\phi}}(t)\Bigg]\end{aligned} \tag{9}$$

where $\widehat{\boldsymbol{f_{\theta^-}}}$ is the parameterized sCM as in Eq. (4) after replacing $\boldsymbol{F_\theta}$ with $\widehat{\boldsymbol{F_\theta}}$ in Proposition 3.1, $\boldsymbol{x}_t, t$ refers to $\boldsymbol{x}_{t,\texttt{Trig}}, t_{\texttt{Trig}}$, and $w_{\boldsymbol{\phi}}(t)$ is an adaptive weighting function to minimize variance across different timesteps following [19, 34].

3.2. Stabilizing Continuous-Time Distillation

To stabilize continuous-time consistency distillation, we address two key challenges: training instabilities and excessively large gradient norms that occur when scaling up the model size and increasing resolution, leading to model collapse. We achieve this by refining the time-embedding to be denser and integrating QK-Normalization into self- and cross-attention mechanisms. These modifications enable efficient training and improve stability, allowing for robust performance at higher resolutions and larger model sizes.



(a) QK-Norm vs Grad. (b) Time embedding vs Grad. (c) Timestep Embedding Similarity

Figure 3. **Efficient Distillation via QK Normalization, Dense Timestep Embedding, and Training-free Schedule Transformation.** (a) We compare gradient norms and visualizations with/without QK Normalization, showing its stabilizing effect. (b) Gradient norm curves for timestep scales (0∼1 vs. 0∼1000) highlight impacts on stability and stability and quality. (c) PCA-based similarity analysis of timestep embeddings. (d) Image results after 5,000 iterations of fine-tuning with (left) and without (right) the proposed schedule transfer (Sec. 3.1).

Dense Time-Embedding. As analyzed in sCM [34], the instability issues in continuous-time CMs primarily stem from the unstable scale of $\frac{\mathrm{d}\boldsymbol{f_\theta}}{\mathrm{d}t}$ in Eq. (9). This instability can be traced back to the expression $\frac{\mathrm{d}F_{\theta^-}}{\mathrm{d}t} = \nabla_{\boldsymbol{x}_t} F_{\theta^-} \frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} + \partial_t F_{\theta^-}$ in Eq. (5), which ultimately originates from the time derivative term $\partial_t F_{\theta^-}$:

$$\partial_t F_{\theta^-} = \frac{\partial F_{\theta^-}}{\partial \mathrm{emb}(c_{\mathrm{noise}})} \cdot \frac{\partial \mathrm{emb}(c_{\mathrm{noise}})}{\partial c_{\mathrm{noise}}} \cdot \frac{\boldsymbol{\partial c_{\mathrm{noise}}(t)}}{\boldsymbol{\partial t}} \tag{10}$$

In previous flow matching models like SANA [63], SD3 [9], and FLUX [23], the noise coefficient $c_{\mathrm{noise}}(t) = 1000t$ amplifies the time derivative $\partial_t F_{\theta^-}$ by a factor of 1000, leading to significant training fluctuations. To address this, we set $c_{\mathrm{noise}}(t) = t$ and fine-tuned SANA for 5k iterations. As shown in Fig. 3 (b), this adjustment reduce excessively large gradient norms (originally exceeding 10^3) to more stable levels. Furthermore, PCA visualization in Fig. 3 (c) reveals that our dense time-embedding design results in more densely packed and similar embeddings for timesteps between 0∼1. This refinement improved training stability and accelerated convergence over 15k iterations.

QK-Normalization. When scaling up the model from 0.6B to 1.6B, we encounter similar issues with excessively large gradient norms, often exceeding 10^3, which lead to training collapse. To address this, we introduce RMS normalization [71] to the Query and Key in both self- and cross-attention modules of the teacher model during fine-tuning. This modification enhances training stability significantly, even with a brief fine-tuning process of only 5,000 iterations. By using the fine-tuned teacher model to initialize the student model, we achieve a more stable gradient norm, as shown in Fig. 3 (a), thereby making the distillation process viable where it was previously infeasible.

3.3. Improving Continuous-Time CMs with GAN

CTM [20] analyzes that CMs distill teacher information in a local manner, where at each iteration, the student model learns from local time intervals. This leads the model to learn cross timestep information under the implicit extrapolation, which can slow the convergence speed. To address this limitation, we introduce an additional adversarial loss [51] to provide direct global supervision across different timesteps, improving both the convergence speed and the output quality.

GANs [12] consist of a generator G and a discriminator D that compete in a zero-sum game to produce realistic synthetic data. Diffusion-GANs [61] and LADD [51] extend this framework by enabling the discriminator to distinguish between noisy real and fake samples. Furthermore, LADD introduces a novel approach by utilizing a frozen teacher model as a feature extractor and training multiple discriminator heads on the teacher model. This methodology facilitates direct adversarial supervision in the latent space, as opposed to the traditional pixel space, leading to more efficient and effective training. Following LADD, we use a hinge loss [28] to train the student model and discriminator

$$\begin{aligned}&\mathcal{L}_{\mathrm{adv}}^{G}(\boldsymbol{\theta})\\&= -\mathbb{E}_{\boldsymbol{x}_0,s,t}\Big[\sum_k D_{\boldsymbol{\psi},k}(\boldsymbol{F}_{\boldsymbol{\theta}^{\mathrm{pre}},k}(\hat{\boldsymbol{x}}_s^{\boldsymbol{f_\theta}}, s, \boldsymbol{y}))\Big],\end{aligned} \tag{11}$$

$$\begin{aligned}&\mathcal{L}_{\mathrm{adv}}^{D}(\boldsymbol{\psi})\\&= \mathbb{E}_{\boldsymbol{x}_0,s}\Big[\sum_k \mathrm{ReLU}\Big(1 - D_{\boldsymbol{\psi},k}(\boldsymbol{F}_{\boldsymbol{\theta}^{\mathrm{pre}},k}(\boldsymbol{x}_s, s, \boldsymbol{y}))\Big)\Big]\\&+ \mathbb{E}_{\boldsymbol{x}_0,s,t}\Big[\sum_k \mathrm{ReLU}\Big(1 + D_{\boldsymbol{\psi},k}(\boldsymbol{F}_{\boldsymbol{\theta}^{\mathrm{pre}},k}(\hat{\boldsymbol{x}}_s^{\boldsymbol{f}_{\boldsymbol{\theta}^-}}, s, \boldsymbol{y}))\Big)\Big],\end{aligned} \tag{12}$$

where $\boldsymbol{x}_s, \hat{\boldsymbol{x}}_s^{\boldsymbol{f_\theta}}, \hat{\boldsymbol{x}}_s^{\boldsymbol{f}_{\boldsymbol{\theta}^-}}$ are the noisy versions of $\boldsymbol{x}_0, \hat{\boldsymbol{x}}_0^{\boldsymbol{f_\theta}} := \boldsymbol{f_\theta}(\boldsymbol{x}_t, t, \boldsymbol{y}), \hat{\boldsymbol{x}}_0^{\boldsymbol{f}_{\boldsymbol{\theta}^-}} := \boldsymbol{f}_{\boldsymbol{\theta}^-}(\boldsymbol{x}_t, t, \boldsymbol{y})$.

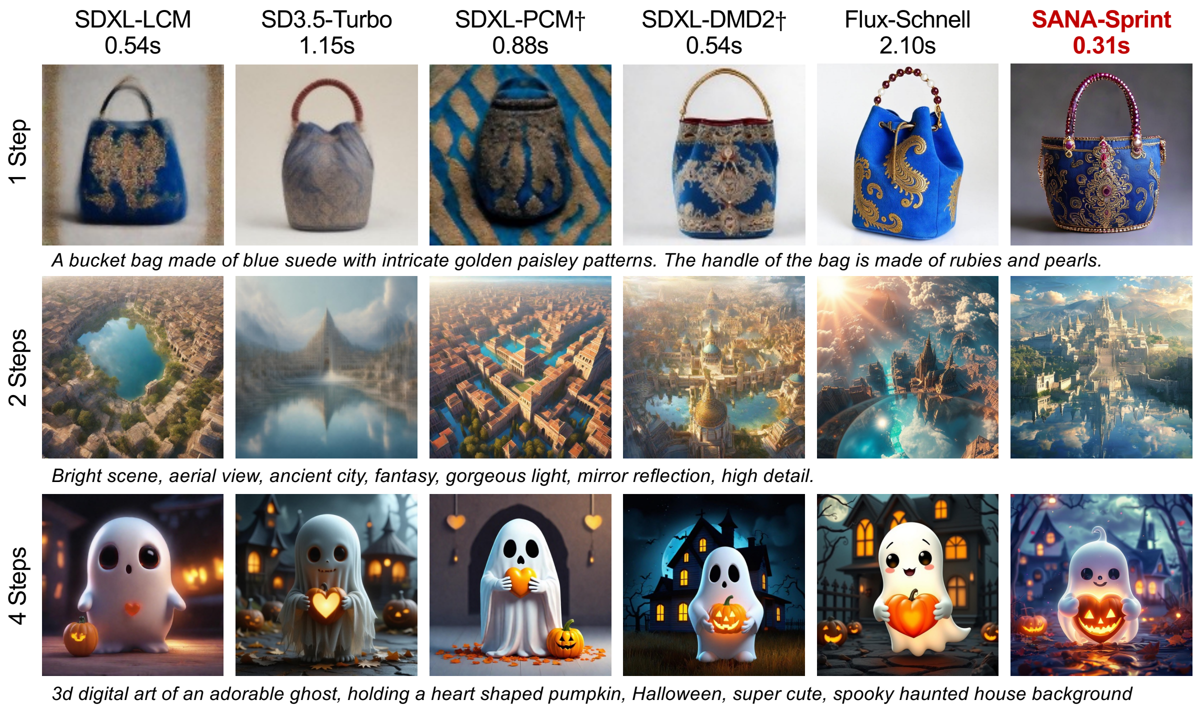


Figure 4. **Visual comparison among SANA-Sprint and selected competing methods in different inference steps.** † indicates that distinct models are required for different inference steps, and time below the method name is the latency of 4 steps tested on A100 GPU. SANA-Sprint produces images with superior realism and text alignment in all inference steps with the fastest speed.

The adversarial loss $\mathcal{L}_{\text{adv}}$ is equivalent to the GAN loss shown at the bottom of Fig. 2. In summary, SANA-Sprint combines sCM loss with GAN loss: $\mathcal{L} = \mathcal{L}_{sCM} + \lambda \mathcal{L}_{\text{adv}}$, where $\lambda = 0.5$ by default, as in Tab. 5.

Additional Max-Time Weighting. In our early experiments, we adopt the timestep sampling distribution of sCM's generator (student model) for GAN loss, given by $t = \arctan\left(\frac{e^{\tau}}{\sigma_d}\right)$, where $\tau \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2)$ with two hyperparameters P_{mean} and P_{std}. To further enhance one- and few-step generation performance and improve overall generation quality, we introduce an additional weighting at $t = \frac{\pi}{2}$. Specifically, with probability p, the training timestep is set to $\frac{\pi}{2}$, while with probability $1 - p$, it follows the original timestep sampling distribution of sCM's generator. We find that this modification significantly improves the model's capability for one- and few-step generation, as shown in Tab. 6.

3.4. Application: Real-Time Interactive Generation

Extending the SANA-Sprint to image-to-image tasks is straightforward. We apply the SANA-Sprint training pipeline to ControlNet [72] tasks, which utilize both images and prompts as instructions. Our approach involves continuing the training of a pre-trained text-to-image diffusion model with a diffusion objective on a dataset adjusted for ControlNet tasks, resulting in the SANA-ControlNet model. We then distill this model using the SANA-Sprint framework to obtain SANA-Sprint-ControlNet.

For ControlNet tasks, we extract Holistically-Nested Edge Detection (HED) scribbles from input images as conditions to guide image generation. Following PixArt's [7] design principles, we train our SANA-ControlNet teacher model on 1024×1024 resolution images. During sampling, HED maps serve as additional conditioning inputs to the Transformer model, allowing precise control over image generation while maintaining structural details. Our experiments show that the distilled SANA-Sprint-ControlNet model retains the controllability of the teacher model and achieves fast inference speeds of approximately 200 ms on H100 machines, enabling near-real-time interaction. The effectiveness of our approach is demonstrated in Appendix F.3.

4. Experiments

4.1. Experimental Setup

Our experiments employ a two-phase training strategy, with detailed settings and evaluation protocols outlined in Appendix F.1. The teacher models are pruned and fine-tuned from the larger SANA-1.5 4.8B model [64], followed by distillation using our proposed training paradigm. We evaluate performance using metrics including FID, CLIP Score on the MJHQ-30K [24], and GenEval [11].

4.2. Efficiency and Performance Comparison

We compare SANA-Sprint with state-of-the-art text-to-image diffusion and timestep distillation methods in Tab. 2

Table 2. **Comprehensive comparison of SANA-Sprint with SOTA approaches in efficiency and performance.** The speed is tested on one A100 GPU with BF16 Precision. Throughput: Measured with batch=10. Latency: Measured with batch=1. We highlight the **best** and second best entries. † indicates that distinct models are required for different inference steps.

	Methods	Inference steps	Throughput (samples/s)	Latency (s)	Params (B)	FID ↓	CLIP ↑	GenEval ↑
Pre-train Models	SDXL [45]	50	0.15	6.5	2.6	6.63	29.03	0.55
	PixArt-Σ [5]	20	0.4	2.7	0.6	6.15	28.26	0.54
	SD3-medium [10]	28	0.28	4.4	2.0	11.92	27.83	0.62
	FLUX-dev [23]	50	0.04	23.0	12.0	10.15	27.47	0.67
	Playground v3 [30]	-	0.06	15.0	24	-	-	0.76
	SANA 0.6B [63]	20	1.7	0.9	0.6	5.81	28.36	0.64
	SANA 1.6B [63]	20	1.0	1.2	1.6	5.76	28.67	0.66
Distillation Models	SDXL-LCM [36]	4	2.27	0.54	0.9	10.81	28.10	0.53
	PixArt-LCM [7]	4	2.61	0.50	0.6	8.63	27.40	0.44
	PCM [60]†	4	1.95	0.88	0.9	15.55	27.53	0.56
	SD3.5-Turbo [9]	4	0.94	1.15	8.0	11.97	27.35	0.72
	SDXL-DMD2 [69]†	4	2.27	0.54	0.9	*6.82*	**28.84**	0.60
	FLUX-schnell [23]	4	0.5	2.10	12.0	7.94	*28.14*	*0.71*
	SANA-Sprint 0.6B	4	5.34	0.32	0.6	**6.48**	28.45	0.76
	SANA-Sprint 1.6B	4	5.20	0.31	1.6	6.54	28.45	**0.77**
	SDXL-LCM [36]	2	2.89	0.40	0.9	18.11	27.51	0.44
	PixArt-LCM [7]	2	3.52	0.31	0.6	10.33	27.24	0.42
	SD3.5-Turbo [9]	2	1.61	0.68	8.0	51.47	25.59	0.53
	PCM [60]†	2	2.62	0.56	0.9	14.70	27.66	0.55
	SDXL-DMD2 [69]†	2	2.89	0.40	0.9	*7.61*	**28.87**	0.58
	FLUX-schnell [23]	2	0.92	1.15	12.0	7.75	28.25	*0.71*
	SANA-Sprint 0.6B	2	6.46	0.25	0.6	6.54	*28.40*	0.76
	SANA-Sprint 1.6B	2	5.68	0.24	1.6	**6.50**	28.45	**0.77**
	SDXL-LCM [36]	1	3.36	0.32	0.9	50.51	24.45	0.28
	PixArt-LCM [7]	1	4.26	0.25	0.6	73.35	23.99	0.41
	PixArt-DMD [5]†	1	4.26	0.25	0.6	9.59	26.98	0.45
	SD3.5-Turbo [9]	1	2.48	0.45	8.0	52.40	25.40	0.51
	PCM [60]†	1	3.16	0.40	0.9	30.11	26.47	0.42
	SDXL-DMD2 [69]†	1	3.36	0.32	0.9	7.10	**28.93**	0.59
	FLUX-schnell [23]	1	1.58	0.68	12.0	*7.26*	28.49	*0.69*
	SANA-Sprint 0.6B	1	7.22	0.21	0.6	**7.04**	28.04	0.72
	SANA-Sprint 1.6B	1	6.71	0.21	1.6	7.69	*28.27*	**0.76**

and Fig. 4. Our SANA-Sprint models focus on timestep distillation, achieving high-quality generation with 1-4 inference steps, competing with the 20-step teacher model, as shown in Fig. 5. More details about the timestep setting are given in Appendix F.2.

Specifically, with 4 steps, SANA-Sprint 0.6B achieves 5.34 samples/s throughput and 0.32s latency, with an FID of 6.48 and GenEval of 0.76. SANA-Sprint 1.6B has slightly lower throughput (5.20 samples/s) but improves GenEval to 0.77, outperforming larger models like FLUX-schnell (12B), which achieves only 0.5 samples/s with 2.10s latency. At 2 steps, SANA-Sprint models remain efficient: SANA-Sprint 0.6B reaches 6.46 samples/s with 0.25s latency (FID: 6.54), while SANA-Sprint 1.6B achieves 5.68 samples/s with 0.24s latency (FID: 6.76). In single-step mode, SANA-Sprint 0.6B achieves 7.22 samples/s throughput and 0.21s latency, maintaining an FID of 7.04 and GenEval of 0.72, comparable to FLUX-schnell but with significantly higher efficiency.

These results demonstrate the practicality of SANA-Sprint for real-time applications, combining fast inference speeds with strong performance metrics.

4.3. Analysis

In this section, we apply a 2-step sampling starting at $t_{max} = \pi/2$ with an intermediate step $t = 1.0$.

Schedule Transfer. To validate the effectiveness of our proposed schedule transfer in Sec. 3.1, we conduct ablation studies on a flow matching model SANA [63], comparing its performance with and without schedule transformation to TrigFlow [34]. As shown in Fig. 3 (d), removing schedule transfer leads to training divergence due to incorrect signals. In contrast, incorporating our schedule transfer enables

Table 3. Comparison of loss combination.

sCM	LADD	FID ↓	CLIP ↑
✓		8.93	27.51
	✓	12.20	27.00
✓	✓	8.11	28.02

Table 4. Comparison of CFG training strategies.

CFG Setting	FID ↓	CLIP ↑
w/o Embed	9.23	27.15
w/ Embed	8.72	28.09

Table 5. sCM and LADD loss weighting.

sCM:LADD	FID ↓	CLIP ↑
1.0:1.0	8.81	27.93
1.0:0.5	8.43	27.85
1.0:0.1	8.90	27.76

Table 6. Comparison of max-time weighting strategy.

Max-Time	FID ↓	CLIP ↑
0% maxT	9.44	27.65
50% maxT	8.32	27.94
70% maxT	8.11	28.02

the model to achieve decent results within 5,000 iterations, demonstrating its crucial role in efficiently adapting flow matching models to TrigFlow-based consistency models.

Influence of CFG Embedding. To clarify the influence of Classifier-Free Guidance (CFG) embedding in our model, we maintain the setting of incorporating CFG into the teacher model, as established in previous works [14, 34, 36]. Specifically, during the training of the student model, we uniformly sample the CFG scale of the teacher model from the set $4.0, 4.5, 5.0$. To integrate CFG embedding [39] into the student model, we add it as an additional condition to the time embedding, multiplying the CFG scale by 0.1 to align with our denser timestep embeddings. We conduct experiments with and without CFG embedding to evaluate its role. As shown in Tab. 4, incorporating CFG embedding significantly improves CLIP score by 0.94.

Effects of sCM and LADD. We evaluate the effectiveness of each component by comparing models trained with only the sCM loss or the LADD loss. As shown in Tab. 3, training with LADD alone results in instability and suboptimal performance, achieving a higher FID score of 12.20 and a lower CLIP score of 27.00. In contrast, combining both sCM and LADD losses improves model performance, yielding a lower FID score of 8.11 and a higher CLIP score of 28.02, demonstrating their complementary benefits. Using sCM alone achieves a FID score of 8.93 and a CLIP score of 27.51, indicating that while sCM is effective, adding LADD further enhances performance. The weighting ablations for sCM and LADD loss are shown in Tab. 5, with additional timestep distribution ablations provided in Appendix F.2

Additional Max-Time Weighting. We validate the proposed max-time weighting strategy in LADD (see Sec. 3.3) through experiments with both sCM and LADD losses. As shown in Tab. 6, this weighting significantly improves performance. We test the strategy at 0%, 50%, and 70% max-time ($t = \pi/2$) probabilities, finding that 50% is the best balance, while higher probabilities provide only marginal gains. However, considering the qualitative results, we finally choose 50% as the default max-time weighting.

5. Related Work

We put a relatively brief overview of related work here, with a more comprehensive version in the appendix. Diffusion models have two primary paradigms for step distillation: trajectory-based and distribution-based methods. Trajectory-based approaches include direct distillation[35] and progressive distillation[39, 49]. Consistency models [54] include variants like LCM [36], CTM [20], MCM [13], PCM [60], and sCM [34]. Distribution-based methods involve GAN-based distillation [12] and VSD variants [37, 46, 50, 62, 66]. Recent improvements include adversarial training with DINOv2 [41][52], stabilization of VSD[70], and improved algorithms like SID [75] and SIM [38]. In real-time image generation, techniques like PaGoDA[21] and Imagine-Flash accelerate diffusion inference. Model compression strategies include BitsFusion[55] and Weight Dilation[32]. Mobile applications use MobileDiffusion[74], SnapFusion[26], and SnapGen[16]. SVDQuant[25] combined with SANA[63] enables fast image generation on consumer GPUs.

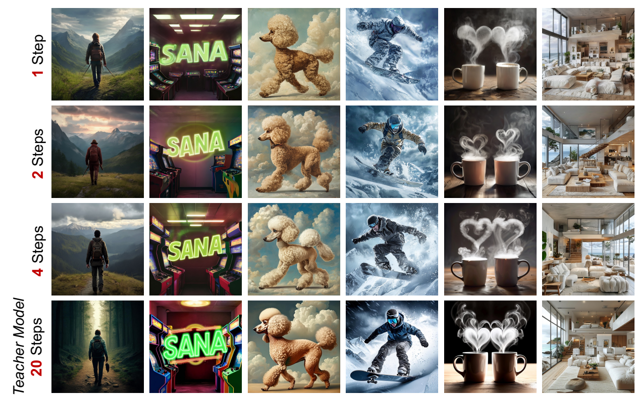


Figure 5. **Visual comparison among SANA-Sprint with different inference steps and the teacher model SANA.** SANA-Sprint can generate high-quality images with one or two steps and the images can be better when increasing steps.

6. Conclusion

In this paper, we introduced SANA-Sprint, an efficient diffusion model for ultra-fast one-step text-to-image generation while preserving multi-step sampling flexibility. By employing a hybrid distillation strategy combining continuous-time consistency distillation (sCM) and latent adversarial distillation (LADD), SANA-Sprint achieves SoTA performance with 7.04 FID and 0.72 GenEval in one step, eliminating step-specific training. This unified step-adaptive model enables high-quality 1024×1024 image generation in only 0.1s on H100, setting a new SoTA in speed-quality tradeoffs.

Looking ahead, SANA-Sprint's instant feedback unlocks real-time interactive applications, transforming diffusion models into responsive creative tools and AIPC. We will open-source our code and models to encourage further exploration in efficient, practical generative AI systems.

References

[1] Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022. 3

[2] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022. 3

[3] Han Cai, Muyang Li, Qinsheng Zhang, Ming-Yu Liu, and Song Han. Condition-aware neural network for controlled image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7194–7203, 2024. 4

[4] Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. *arXiv preprint arXiv:2410.10733*, 2024. 4

[5] Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-σ: Weak-to-strong training of diffusion transformer for 4k text-to-image generation. *arXiv preprint arXiv:2403.04692*, 2024. 7, 4

[6] Junsong Chen, YU Jincheng, GE Chongjian, Lewei Yao, Enze Xie, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *International Conference on Learning Representations*, 2024. 4

[7] Junsong Chen, Yue Wu, Simian Luo, Enze Xie, Sayak Paul, Ping Luo, Hang Zhao, and Zhenguo Li. Pixart-{\delta}: Fast and controllable image generation with latent consistency models. *arXiv preprint arXiv:2401.05252*, 2024. 6, 7

[8] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 4

[9] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024. 2, 5, 7, 4

[10] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024. 7, 4

[11] Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36:52132–52152, 2023. 6, 5

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 1, 5, 8, 4

[13] Jonathan Heek, Emiel Hoogeboom, and Tim Salimans. Multistep consistency models. *arXiv preprint arXiv:2403.06807*, 2024. 1, 8, 4

[14] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 8

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1, 2

[16] Dongting Hu, Jierun Chen, Xijie Huang, Huseyin Coskun, Arpit Sahni, Aarush Gupta, Anujraaj Goyal, Dishani Lahiri, Rajesh Singh, Yerlan Idelbayev, et al. Snapgen: Taming high-resolution text-to-image models for mobile devices with efficient architectures and training. *arXiv preprint arXiv:2412.09619*, 2024. 8, 4

[17] 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. In *European Conference on Computer Vision*, pages 428–447. Springer, 2024. 1

[18] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022. 3, 4, 5

[19] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24174–24184, 2024. 4

[20] 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. 1, 5, 8, 4

[21] Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Yuhta Takida, Naoki Murata, Toshimitsu Uesaka, Yuki Mitsufuji, and Stefano Ermon. Pagoda: Progressive growing of a one-step generator from a low-resolution diffusion teacher. *Advances in Neural Information Processing Systems*, 37:19167–19208, 2025. 8, 4

[22] Jonas Kohler, Albert Pumarola, Edgar Schönfeld, Artsiom Sanakoyeu, Roshan Sumbaly, Peter Vajda, and Ali Thabet. Imagine flash: Accelerating emu diffusion models with backward distillation. *arXiv preprint arXiv:2405.05224*, 2024.

[23] Black Forest Labs. Flux, 2024. 2, 5, 7, 4

[24] Daiqing Li, Aleks Kamko, Ehsan Akhgari, Ali Sabet, Linmiao Xu, and Suhail Doshi. Playground v2. 5: Three insights towards enhancing aesthetic quality in text-to-image generation. *arXiv preprint arXiv:2402.17245*, 2024. 6, 5

[25] Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*, 2024. 8, 4

[26] Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. *Advances in Neural Information Processing Systems*, 36:20662–20678, 2023. 8, 4

[27] Zhimin Li, Jianwei Zhang, Qin Lin, Jiangfeng Xiong, Yanxin Long, Xinchi Deng, Yingfang Zhang, Xingchao Liu, Minbin Huang, Zedong Xiao, et al. Hunyuan-dit: A powerful multi-resolution diffusion transformer with fine-grained chinese understanding. *arXiv preprint arXiv:2405.08748*, 2024.

[28] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017. 5

[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. 2

[30] Bingchen Liu, Ehsan Akhgari, Alexander Visheratin, Aleks Kamko, Linmiao Xu, Shivam Shrirao, Joao Souza, Suhail Doshi, and Daiqing Li. Playground v3: Improving text-to-image alignment with deep-fusion large language models. *arXiv preprint arXiv:2409.10695*, 2024. 7, 4

[31] 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. 2

[32] Xuewen Liu, Zhikai Li, and Qingyi Gu. Dilatequant: Accurate and efficient diffusion quantization via weight dilation. *arXiv preprint arXiv:2409.14307*, 2024. 8, 4

[33] Cheng Lu. Research on reversible generative models and their efficient algorithms, 2023.

[34] Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. *arXiv preprint arXiv:2410.11081*, 2024. 1, 3, 4, 5, 7, 8

[35] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021. 1, 8, 4

[36] 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. 1, 3, 7, 8, 4

[37] Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. *Advances in Neural Information Processing Systems*, 36:76525–76546, 2023. 1, 8, 4

[38] Weijian Luo, Zemin Huang, Zhengyang Geng, J Zico Kolter, and Guo-jun Qi. One-step diffusion distillation through score implicit matching. *Advances in Neural Information Processing Systems*, 37:115377–115408, 2025. 8, 4

[39] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14297–14306, 2023. 1, 8, 4

[40] OpenAI. Dalle-3, 2023.

[41] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 8, 4

[42] William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022. 4

[43] Stefano Peluchetti. Non-denoising forward-time diffusions, 2022. 2

[44] Pablo Pernias, Dominic Rampas, Mats L. Richter, Christopher J. Pal, and Marc Aubreville. Wuerstchen: An efficient architecture for large-scale text-to-image diffusion models, 2023. 4

[45] 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. 7

[46] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 1, 8, 4

[47] Jingjing Ren, Wenbo Li, Haoyu Chen, Renjing Pei, Bin Shao, Yong Guo, Long Peng, Fenglong Song, and Lei Zhu. Ultrapixel: Advancing ultra-high-resolution image synthesis to new peaks. *arXiv preprint arXiv:2407.02158*, 2024. 4

[48] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 4

[49] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 1, 8, 4

[50] Tim Salimans, Thomas Mensink, Jonathan Heek, and Emiel Hoogeboom. Multistep distillation of diffusion models via moment matching. *Advances in Neural Information Processing Systems*, 37:36046–36070, 2025. 8, 4

[51] Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024. 1, 5, 4

[52] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pages 87–103. Springer, 2024. 1, 8, 4

[53] 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. 1, 2

[54] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023. 1, 3, 8, 4

[55] Yang Sui, Yanyu Li, Anil Kag, Yerlan Idelbayev, Junli Cao, Ju Hu, Dhritiman Sagar, Bo Yuan, Sergey Tulyakov, and Jian Ren. Bitsfusion: 1.99 bits weight quantization of diffusion model. *arXiv preprint arXiv:2406.04333*, 2024. 8, 4

[56] Haotian Tang, Yecheng Wu, Shang Yang, Enze Xie, Junsong Chen, Junyu Chen, Zhuoyang Zhang, Han Cai, Yao Lu, and Song Han. Hart: Efficient visual generation with hybrid autoregressive transformer. *arXiv preprint arXiv:2410.10812*, 2024.

[57] Yao Teng, Yue Wu, Han Shi, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Dim: Diffusion mamba for efficient high-resolution image synthesis. *arXiv preprint arXiv:2405.14224*, 2024. 4

[58] Yuchuan Tian, Zhijun Tu, Hanting Chen, Jie Hu, Chao Xu, and Yunhe Wang. U-dits: Downsample tokens in u-shaped

diffusion transformers. *arXiv preprint arXiv:2405.02730*, 2024. 4

[59] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011. 2

[60] Fu-Yun Wang, Zhaoyang Huang, Alexander William Bergman, Dazhong Shen, Peng Gao, Michael Lingelbach, Keqiang Sun, Weikang Bian, Guanglu Song, Yu Liu, et al. Phased consistency model. *arXiv preprint arXiv:2405.18407*, 2024. 1, 7, 8, 4

[61] Zhendong Wang, Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Diffusion-gan: Training gans with diffusion. *arXiv preprint arXiv:2206.02262*, 2022. 5

[62] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36:8406–8441, 2023. 1, 8, 4

[63] Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, et al. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. *arXiv preprint arXiv:2410.10629*, 2024. 5, 7, 8, 4

[64] Enze Xie, Junsong Chen, Yuyang Zhao, Jincheng Yu, Ligeng Zhu, Yujun Lin, Zhekai Zhang, Muyang Li, Junyu Chen, Han Cai, et al. Sana 1.5: Efficient scaling of training-time and inference-time compute in linear diffusion transformer. *arXiv preprint arXiv:2501.18427*, 2025. 6, 5

[65] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.

[66] Sirui Xie, Zhisheng Xiao, Diederik P Kingma, Tingbo Hou, Ying Nian Wu, Kevin Patrick Murphy, Tim Salimans, Ben Poole, and Ruiqi Gao. Em distillation for one-step diffusion models. *arXiv preprint arXiv:2405.16852*, 2024. 8, 4

[67] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36, 2024.

[68] Jing Nathan Yan, Jiatao Gu, and Alexander M Rush. Diffusion models without attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8239–8249, 2024. 4

[69] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. *arXiv preprint arXiv:2405.14867*, 2024. 1, 7

[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. 8, 4

[71] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019. 5

[72] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847, 2023. 6

[73] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 3

[74] Yang Zhao, Yanwu Xu, Zhisheng Xiao, Haolin Jia, and Tingbo Hou. Mobilediffusion: Instant text-to-image generation on mobile devices. In *European Conference on Computer Vision*, pages 225–242. Springer, 2024. 8, 4

[75] Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In *Forty-first International Conference on Machine Learning*, 2024. 8, 4

[76] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 1

[77] Lianghui Zhu, Zilong Huang, Bencheng Liao, Jun Hao Liew, Hanshu Yan, Jiashi Feng, and Xinggang Wang. Dig: Scalable and efficient diffusion models with gated linear attention. *arXiv preprint arXiv:2405.18428*, 2024. 4

[78] 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.