SD3.5-Flash: Distribution-Guided Distillation of Generative Flows

Hmrishav Bandyopadhyay[1,2], Rahim Entezari[1], Jim Scott[1], Reshinth Adithyan[1], Yi-Zhe Song[2], and Varun Jampani[1]

[1] Stability AI
[2] SketchX, CVSSP, University of Surrey

Fig. 1: First Look: High-fidelity samples (prompts and more samples in supplementary) from our 4-step model demonstrate exceptional prompt adherence and compositional understanding

Abstract. We present SD3.5-Flash, an efficient few-step distillation framework that brings high-quality image generation to accessible consumer devices. Our approach distills computationally prohibitive rectified flow models through a reformulated distribution matching objective tailored specifically for few-step generation. We introduce two key innovations: "timestep sharing" to reduce gradient noise and "split-timestep fine-tuning" to improve prompt alignment. Combined with comprehensive pipeline optimizations like text encoder restructuring and specialized quantization, our system enables both rapid generation and memory-efficient deployment across different hardware configurations. This democratizes access across the full spectrum of devices, from mobile phones to desktop

computers. Through extensive evaluation including large-scale user studies, we demonstrate that SD3.5-Flash consistently outperforms existing few-step methods, making advanced generative AI truly accessible for practical deployment.

1 Introduction

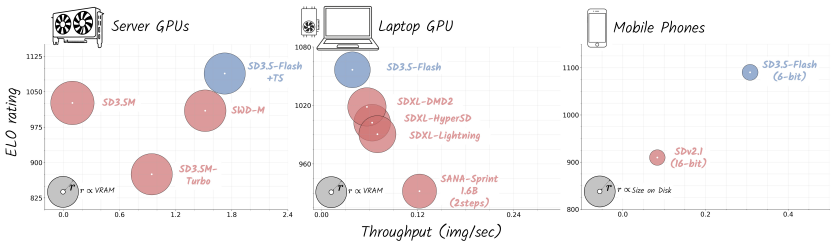

Fig. 2: SD3.5-Flash Suite (right): We introduce the SD3.5-Flash suite of models, preferred by users over all other models at a variety of consumer compute budgets while offering comparable latency and memory requirements. Bubble size indicates VRAM occupied and pipeline size on disk for gpus and mobile devices respectively. We compute Elo ratings by assessing generated image quality via human rankings for different models. **We deploy our 6-bit model on an iPhone and include video demo in supplementary zip.**

Today's best image generation models are trapped in datacenters. While rectified flow models achieve unprecedented quality, their computational demands – 25+ steps, 16GB+ VRAM, 30+ seconds per image – make them inaccessible to everyday devices. We bridge this gap, enabling high-quality image generation in a variety of devices, from mobile phones to gaming desktops.

Timestep distillation offers a path forward in reducing the inference cost of diffusion based image generation. Approaches like distribution matching can reduce step counts in multi-step diffusion inference, but the core challenge emerges from how distribution matching operates in few-step flow distillation. Standard approaches [49,55] require re-noising samples on trajectory end-points to compute distribution divergences through score formulations [47]. This re-noising moves samples off the learned ODE trajectory, where velocity predictions become unreliable and gradient estimates corrupted. In few-step regimes, this problem becomes particularly pronounced as errors cannot be corrected through subsequent iterations, causing systematic quality collapse. Severe capacity constraints imposed by few-step distillation further forces models to sacrifice prompt-image alignment as they struggle to maintain both aesthetic quality and semantic fidelity.

We propose SD3.5-Flash, a few-step rectified flow model that enables high-quality image generation (Fig. 1) on consumer hardware (Fig. 2). To train for improved aesthetic quality with few-step flow distillation, we introduce timestep sharing: computing distribution matching with samples on the student trajectory rather than estimates to random trajectory points. This provides stable gradient signals for known noise levels and reliable flow predictions on the ODE trajectory, improving training stability and consequently model performance.

We next introduce split-timestep fine-tuning which addresses the prompt alignment challenge by temporarily expanding model capacity during training.

Instead of forcing compressed parameters to handle both aesthetic quality and semantic fidelity simultaneously, we branch our model for different timestep ranges. During inference, we merge these branches into a unified checkpoint.

To truly deliver on the "flash" promise, we implement pipeline optimizations extending beyond our core algorithmic innovation. We restructure text encoders as optional (T5-XXL) and necessary (CLIP-L/G) components by exploiting encoder dropout pre-training and quantize from 16-bit to 6-bit precision to balance memory footprint against inference speed. The result is model variants that democratize access across the full spectrum of devices from mobile to desktop, with tailored configurations for each computational tier (Fig. 2).

Our contributions are aimed to improve accessibility to few-step image generation models through: (i) timestep sharing that provides stable gradients by leveraging intermediate trajectory information, (ii) split-timestep fine-tuning that resolves the capacity-quality tradeoff during distillation, and (iii) comprehensive pipeline optimizations that enable practical deployment on a diverse range of commodity hardware. Through extensive evaluation including large-scale user studies, we demonstrate that our approach consistently outperforms existing methods across diverse hardware configurations while maintaining the quality standards of much larger, slower models.

2 Related Works

Diffusion-based generative models [13,32] are inherently slow due to their iterative nature, starting from a base distribution (e.g., Gaussian noise) and gradually denoising it to realistic samples. Skip-step schedulers [44] accelerate diffusion inference by reducing the number of inference timesteps with deterministic sampling [14] while distillation techniques [3, 17, 27, 29, 36] learn a more efficient denoising trajectory from noise to data.

Trajectory preserving distillation techniques distill a multi-step teacher into a few-step student by aligning the student and teacher trajectories [20, 38] and fine-tuning the student to skip steps progressively. The student learns to mimic an approximation of the teacher's trajectory in fewer steps than the teacher. **Progressive Distillation** of this nature, however, cannot learn extreme low-step (*e.g.* two-step) inference [20] due to approximation errors.

Other approaches like discrete [4, 45, 46] and continuous time [5, 26] **Consistency Models** involve learning to jump directly to trajectory endpoints or intermediate points [15, 36] using a more efficient path from noise to data. This improves one-step inference quality while supporting iterative refinement of generated samples through a self-consistency property. Alternately, recent works inspired by Score Distillation Sampling [33, 52], train the student network by **Score Matching** [47] for aligning teacher and student distributions [7, 9, 30, 49, 55, 56, 59]. Different from these approaches, Insta-Flow [25] fine-tunes score based generative models in a rectified flow setting for efficient inference. SWD [49] applies DMD for scale wise distillation in a rectified flow setup.

Approaches like progressive distillation, consistency distillation, and score matching are generally unstable or inadequate by themselves and have been supplemented with adversarial techniques in recent works like SDXL-Lightning [20],

Hyper-SD [36] and DMD-2 [55]. This adversarial objective is generally optimized by comparing fake samples generated by the few-step student with real [55] or synthetic samples [39] from the multi-step teacher in a generator discriminator setting. Recent work [20, 39] also reformulates this GAN setup to use the teacher as a discriminative feature extractor, for enhancing discriminator quality at no additional cost. This allows for adding multiple lightweight discriminator heads [3] to construct multi-discriminator setups [40, 42] which offer richer generator updates and training stability through diverse adversarial feedback in GANs. Nitrofusion [3] demonstrates that multi-discriminator adversarial setups are enough without supplementary objectives for stable one-step distillation from low-step models.

Orthogonal to distillation, some methods look to reduce diffusion model parameters [6, 19, 23, 58] to further bring down inference cost both in terms of speed and compute. Since attention units take up a large chunk of compute, particularly in recent Diffusion Transformer (DiT) architectures, a majority of works focus on removing [58] or replacing [23] them with more efficient alternatives. Separate from the diffusion model itself, the generation pipeline involves the text encoder [34, 35] for conditional context and the VAE [16] for decoding latent space samples to image space. Some works [2, 58] also focus on optimizing the VAE based latent decoding (denoised latent $\rightarrow$ image) by replacing the VAE with a lighter and more efficient decoders.

3 Background

Flow matching: Diffusion Models [13, 32, 37] are a family of generative models that learn a (Gaussian) noise to data trajectory and iteratively follow it to generate media with sampled noise. This trajectory from noise to data is typically modelled as the solution to a Stochastic Differential Equation (SDE) in score-based generative frameworks [44], and can be reformulated as an Ordinary Differential Equation (ODE) known as the probability flow ODE (PF-ODE in [14, 47]). Diffusion models in score based generative frameworks learn a score function — the gradient of the log probability density — by training a neural network to estimate it at various noise levels along the trajectory. The update direction can be defined as :

$$\mathrm{d}x_t = \left[\mu(x_t, t) - \frac{1}{2}\sigma(t)^2 \nabla \log p_t(x_t)\right] \mathrm{d}t \tag{1}$$

where $\nabla \log p_t(x_t)$ is referred to as the score function of $p_t(x_t)$ and is parameterised by a neural network as $s_\theta(x_t, t)$ and in a PF-ODE [14], $\mu(x_t, t) = 0$. In contrast, flow matching [8, 22] models define a separate class of generative methods that directly learn an ODE-based mapping without relying on an underlying SDE. These models parameterise a velocity field that transports samples from noise to data along the ODE-defined trajectory. The update direction with flow matching changes to $\mathrm{d}x_t = v_t(x_t)\mathrm{d}t$ where the velocity $v_t(x_t)$ is parameterised by a network as $v_\theta(x_t, t)$. In rectified flow pipelines [24] like SD3.5 Medium [48], samples are noised following a straight path between the data distribution and standard normal $\mathcal{N}(\mathbf{0}, \mathbf{I})$ as $x_t = (1-t)x_0 + t.\epsilon$

Distribution Matching Distillation: DMD [56] proposes the distillation of a multi-step teacher G into a distilled single-step student G_θ by matching the student distribution p_{fake} with that of the teacher p_{real}. Given a sample $x = G_\theta(z)$ where $z \sim \mathcal{N}(0, \mathbf{I})$ this distribution match is calculated as the Kullback-Leibler (KL) divergence:

$$D_{\text{KL}}(p_{\text{fake}}||p_{\text{real}}) = -\mathbb{E}_{x\sim p_{\text{fake}}}\Big(\log\ \frac{p_{\text{real}}(x)}{p_{\text{fake}}(x)}\Big) \tag{2}$$

However, using this divergence directly as loss is not possible as the probability densities are generally intractable. Since only the gradient of this loss is needed, this can be circumvented, by substituting in score function $s(x) = \nabla_x \log p(x)$ and computing the loss gradient as

$$\nabla_\theta \mathcal{L}_{\text{DMD}} = -\mathbb{E}_{x\sim p_{\text{fake}}}\Big((s_{\text{real}}(x) - s_{\text{fake}}(x))\frac{\mathrm{d}G_\theta}{\mathrm{d}\theta}\Big) \tag{3}$$

To obtain these scores, generated samples x_0 are re-noised up-to timestep t as $x_t = \sqrt{\alpha_t}x + \sqrt{1-\alpha_t}\epsilon$. Then the score is computed from the denoising signal of the pre-trained diffusion models as $s_{\text{real}}(x_t, t)$ for teacher score and $s_{\text{fake}}(x_t, t)$ for student score where $s_{\text{fake}}(x_t, t) = -\frac{x_t - \alpha_t G_\theta(x_t,t)}{\sigma_t^2}$ from the student G_θ. Since the few-step models work only on a subset of timesteps, a multi-step proxy model is maintained that monitors the distribution of the few-step model and acts as a surrogate student score estimator. To stabilise this pipeline, $\mathcal{L}_{\text{DMD}}$ is accompanied by regression loss, calculated as the MSE between images generated by the student and the teacher starting from the same noise. DMD2 [55] proposes updating the student proxy G_ϕ with a biased schedule to improve stability without introducing this regression loss and supplements $\mathcal{L}_{\text{DMD}}$ with an adversarial objective.

4 Methodology

4.1 Trajectory Guidance

For stable pre-training of our 4-step student network, we use a trajectory guidance objective $\mathcal{L}_{\text{TG}}$. For timesteps $t \in [0, 1]$ on the teacher model's trajectory, we subsample points t_i^s which coincide with the student trajectory (*i.e.* $i \in [1, 4]$ for 4-step model) and calculate the trajectory guidance objective as:

$$\mathcal{L}_{\text{TG}} = \sum_i \|t_i^s(G_\theta(x_{t_i^s}, t_i^s) - \int_{t_i^s}^{t_{i-1}^s} v_{\text{real}}(x_t, t)\mathrm{d}t)\|^2 \tag{4}$$

where v_{real} corresponds to the velocity predictor teacher model and G_θ is the student being trained.

4.2 Distribution Matching in Flow Models

To refine our pre-trained student G_θ we use the DMD objective in Eq. (3) that computes the gradient for the KL-divergence between teacher and student distributions with the proxy v_{fake}. Hence, we align the distributions of the proxy and the student, to enable accurate representation of student distribution in $\mathcal{L}_{\text{DMD}}$. This is done by finetuning on generated student samples and computing the standard flow matching objective as $\mathcal{L}_{\text{FM}}$=$||v_{\text{target}} - v_{\text{fake}}(x_t, t)||_2^2$, where v_{target} is from added noise.

Timestep Sharing: The DMD objective in Eq. (3) requires evaluating the real $s_{\text{real}}(x_t, t)$ and fake scores $s_{\text{fake}}(x_t, t)$, at intermediate noise levels t. These intermediate samples are typically obtained by perturbing the clean generated sample x_0 with noise. In stochastic score-based models, this is natural because x_0 is estimated at every denoising step and then re-noised. However, our trajectory-guidance pre-training (Stage (i), optimized with $\mathcal{L}_{\text{TG}}$) is reflow-like: it forms strongly coupled noise-data pairs along a learned ODE trajectory. Once the student follows this trajectory, re-noising a generated sample x_0 to a random timestep with fresh noise moves it off-trajectory, where velocity estimates become unreliable and gradients noisy. We therefore draw intermediate noisy samples directly from the student's own denoising trajectory.

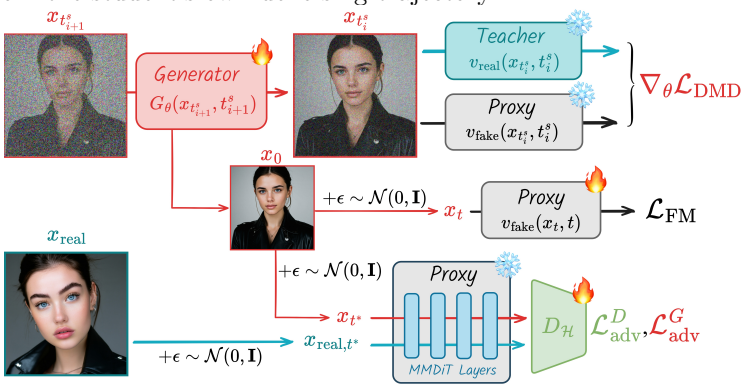

Fig. 3: Training Pipeline: We train G_θ with the distribution matching objective $\nabla_\theta \mathcal{L}_{\text{DMD}}$ and adversarial objective $\mathcal{L}_{\text{adv}}^G$. The proxy student v_{fake} that is used to compute $\nabla_\theta \mathcal{L}_{\text{DMD}}$ is trained with the standard flow matching objective $\mathcal{L}_{\text{FM}}$ and the discriminator for the adversarial objective is trained with $\mathcal{L}_{\text{adv}}^D$.

Specifically, during "backward simulation" in DMD training [55], each training example is constructed by denoising from pure noise at t=1000 using the student G_θ. To train the student at timestep t=500, we first denoise (without gradients) along $x_{1000} \to x_{750} \to x_{500}$ and then compute the velocity v=$G_\theta(x_{500}, t$=$500)$ with gradients enabled. The original DMD formulation would use this velocity to reconstruct x_0, re-noise it to a randomly chosen timestep t, and evaluate the KL divergence in Eq. (3). Instead, we stay on the student's natural trajectory and evaluate the KL divergence using the next noisy state in the trajectory *i.e.* x_{250} here (from $x_{500} \to x_{250}$). This reduces the diversity of timesteps used for DMD estimation but avoids stochastic re-noising entirely. Empirically, we find that this results in improved image composition and overall generation quality (see Sec. 5.5). For the final timestep, whose next step is t=0, we set the DMD objective to zero; this timestep is adequately trained via auxiliary objectives such as the adversarial loss.

Split-Timestep Fine-Tuning: Timestep distillation often weakens the correspondence between text prompts and generated outputs [39] as the model capacity is severely restricted to perform timestep compression while maintaining aesthetic quality. Inspired by the use of diffusion models in muti-task learning [11, 28], we find temporarily relaxing capacity constraints can improve generation aspects

like prompt alignment significantly. For this, we duplicate the partially trained model into branches, M_1 and M_2 and train them on disjoint timestep ranges $t_1 \in (0, 500]$ and $t_2 \in (500, 1000]$ respectively, to increase effective model capacity. During fine-tuning, each branch uses an exponential moving average with a decay of β=0.99 to stabilise and keep weights close to the original checkpoint. After convergence, we fuse the branches by weight interpolation, selecting a $3:7$ ratio $(M_1 : M_2)$ to maximise text-prompt alignment as measured with GenEval [10]. We perform split-timestep fine-tuning during the last stage of training and observe a distinct jump in model performance (see Sec. 5.5).

4.3 Adversarial Loss

Similar to prior works [3, 55], we use an adversarial objective where the proxy student v_{fake} acts as a feature extractor to obtain discriminator features. This allows us to perform adversarial training on the flow latent space as opposed to the image space in [41]. For extracting features using v_{fake}, we noise samples x_0 to pre-defined noise levels at timesteps $t^* \in [0, 1]$ and extract intermediate outputs from $v_{\text{fake}}(x_{t^*}, t^*)$ at multiple layers as feature maps. Timesteps t^* are well distributed in $[0, 1]$ to capture both coarse-grained features ($t^* \approx 1$) and fine-grained features ($t^* \approx 0$). We train MLP discriminator heads $D_{\mathcal{H}}$ on top of these features for real/fake prediction where synthetic samples generated by the teacher model are used as "real" data. Similar to NitroFusion [3], we periodically refresh our discriminator heads by re-initializing their weights to reduce overfitting. We use the standard non saturating GAN objective to train the discriminator heads and the student generator G_θ:

$$\mathcal{L}^D_{\text{adv}} = \mathbb{E}_{x_{t^*} \sim p_{\text{real},t^*}} \log D(x_{t^*}) - \mathbb{E}_{x_{t^*} \sim p_{\text{fake},t^*}} \log D(x_{t^*}) \tag{5}$$

$$\mathcal{L}^G_{\text{adv}} = -\mathbb{E}_{x_{t^*} \sim p_{\text{fake},t^*}} \log D(x_{t^*}) \tag{6}$$

where the discriminator heads $D_{\mathcal{H}}$ (Fig. 3) and the feature extractor are collectively referred to as D.

4.4 Four Step Generation

We start by initializing our teacher, student and the proxy student with pre-trained weights from the multi-step teacher. Next, we perform three stages of training, where we *(i)* pre-train the student model with $\mathcal{L}_{\text{TG}}$ where the model is optimized to replicate the teacher trajectory in few-steps. *(ii)* In the second stage, we minimize the KL divergence of teacher and student distributions $\mathcal{L}_{\text{DMD}}$ supplemented with an adversarial objective from our multi-head discriminator. The first stage of training helps to align teacher and student trajectories and speeds up training of the next stage considerably. The second stage constructs sharp features and detailed images. *(iii)* Lastly, we use split-timestep fine-tuning to temporarily increase model capacity that helps us improve prompt alignment and generation quality during inference.

4.5 Pipeline Optimization

We perform inference optimization on top of the Stable Diffusion 3.5 pipeline. This pipeline consists of three text encoders (CLIP-L [34], CLIP-G [34], and

Table 1: Inference latency: Comparing inference latency of SD3.5-Flash models for different devices with VRAM / unified memory below device names.

Model	Resolution	Latency (in seconds)			
		RTX 4090 24 GB	**M3 MBP** 32 GB	**M4 iPad** 8 GB	**A17 iPhone** 8 GB
SD3.5-Flash 16-bit (w T5-XXL)	1024 px	0.58	18.65	–	–
	768 px	0.34	8.21	–	–
	512 px	0.19	3.74	–	–
SD3.5-Flash 8-bit (w/o T5-XXL)	1024 px	0.61	14.08	–	–
	768 px	0.35	6.32	–	–
	512 px	0.22	2.97	–	–
SD3.5-Flash 6-bit (w/o T5-XXL)	1024 px	–	13.43	–	–
	768 px	–	6.26	6.44	8.32
	512 px	–	3.12	2.62	3.25

T5-XXL [35]) besides the MM-DiT diffusion model [48], and a VAE [16]. Of these, T5-XXL is the largest component, accounting for the bulk of peak VRAM usage and inference time. The full distilled model in 16-bit precision requires 18 GiB of GPU memory—beyond the reach of most consumer cards. To bring this down, we quantize the MM-DiT diffusion model to 8-bit and leverage encoder dropout pre-training in SD3.5 to substitute T5-XXL with null embeddings. This brings our memory requirement down to just about 8 GiB. To truly support edge devices like phones and tablets, we use CoreML on Apple Silicon to quantize our 8-bit model down to 6-bit (Fig. 1). We note that converting inference pipelines to support edge devices is non-trivial and requires re-writing operations to reduce quality degradation. Specifically for this quantization, we rewrite operations like RMSNorm to better preserve precision on the Apple Neural Engine. We summarise the results of our optimzation in Tab. 1, and highlight less than 10s latency on devices like iPhone (**video in supplementary zip**) and iPad. We include more details on memory performance tradeoff in the supplementary.

5 Experiments

5.1 Implementation Details

Dataset and Training: Following previous works [3, 39], we use synthetic samples for training our model as they offer high prompt coherence and are consistent in quality. For our training data, we generate synthetic samples using the SD3.5 Large (8B) model over 32 timesteps and a CFG scale of 4.0. We pre-train for 2K iterations for stage *(i)* (see Sec. 4.4) and then train the 4-step model for 800 iterations in stage *(ii)* and 400 iterations in stage *(iii)*, using the 2.5B SD3.5M as teacher. All training is performed on a single H100 node. We present more training details in supplementary.

Fig. 4: User study: Comparing images generated by SD3.5-Flash with other models.

Baselines: For comparisons, we look at **DMD2** [55], **Hyper-SD** [36], **SDXL-Turbo** [41], **Nitrofusion** [3] and **SDXL-Lightning** that are trained from **SDXL** [32] as the teacher network. DMD2 distils SDXL by matching the distributions of the teacher and the student with the gradient of a KL divergence objective. Hyper-SD performs consistency distillation with trajectory guidance and uses human feedback learning [54] for improving performance. SDXL-Turbo demonstrates adversarial distillation in the rich semantic space of Dino-V2 [31], decoding latents to images throughout training. SDXL-Lightning also uses adversarial distillation, but relaxes mode coverage for the student with a mix of conditional and unconditional objectives in the discriminator. Nitrofusion stabilises adversarial distillation with a multi-discriminator setup and a periodic discriminator refresh, training on SDXL-DMD2 and SDXL-HyperSD. Improving upon SDXL and **SDv2.1** [37], recent models like **SD3.5** [48] and **SANA** [53] offer better generation quality and higher prompt adherence by adopting rectified flow pipelines for faster convergence. **SWD** [49] distils SD3.5M by training a scale wise network, optimized with a distribution matching objective. **SiD-DiT** [59] unifies various distillation objectives through re-weighting to distil SD3.5M along with other models like Flux.1-dev. **SANA-Sprint** [5] uses continuous-time consistency distillation [46] to distil SANA to 1, 2, and 4-step models. We also include comparisons with **SD3.5M-Turbo** released by TensorArt Studios [51] as an stand-alone checkpoint on top of SD3.5M. We do not compare with large models like SD3.5 Large (8B) and Flux.1-dev [1] (12B) which are difficult to fit into consumer grade hardware.

Fig. 5: Removing T5: 4 step quality with and without T5 text encoder. When removing T5, we substitute T5 embeddings with null embeddings. Figures here are generated using complex prompts, which are included in supplementary.

5.2 User Study

We conduct user studies based on image quality and prompt alignment with 178 annotators to evaluate images generated with 4 different seeds. We generate samples using 4 seeds per prompt, from a diverse curated set of prompts consisting of expert-designed prompts and a subset of Parti prompts [57]. For each generated sample, 3 users vote on two images from two different methods, rating them on visual quality and image-prompt correlation (prompt adherence). For a total of 12 competitors in Fig. 4, we run over 54K A v/s B comparisons for image quality alone. From our user studies (in Fig. 4), we find SD3.5-Flash outperforms other few-step models and even our 50 step teacher in image quality. For prompt-adherence, the difference is marginal ($< \pm 1.6\%$) across all methods (more in supplementary). We also compare select competitors against each other to compute Elo scores (see Fig. 1 and Tab. 2). In all compute scenarios our models appear on the top of the Elo ladder demonstrating high quality image generation across a variety of compute budgets. We note that Elo scores for mobile devices

Fig. 6: Qualitative comparisons: Comparing few-step text-to-image generation.

are redundant since there are only two competitors, yet we include this for completeness as it demonstrates that our 6-bit model outperforms SDv2.1 [37].

5.3 Qualitative Comparisons

We include qualitative comparisons of our model (**SD3.5-Flash 16-bit + T5**) with other few-step generation pipelines like SANA-Sprint1.6B, NitroFusion, SDXL-DMD2 and SDXL-Lightning in Fig. 6, and additional comparisons (including SWD) in the supplementary. 4-step results from SDXL-DMD2 [55], SDXL-Lightning [20] and NitroFusion [3] show poor prompt alignment and composition in complex prompts involving human interaction. SDXL-Lightning [20] generates smooth images lacking sharpness and low in detail, and sometimes generates artifacts (*e.g.* two corgis on sofa in last row, last column). SDXL-

Table 2: Elo Ratings: Ratings for models in different inference environments. Models that use SD3.5M are coloured in green.

Model	Elo	Inference Environment
SD3.5-Flash 16-bit(+T5)	**1088.7**	Server Compute (> 12GB VRAM)
SD3.5M [8]	1026.3	
SWD-M [49]	1009.8	
SD3.5M-Turbo [51]	875.2	
SD3.5-Flash 16-bit (No T5)	**1056.6**	Laptop Compute (8–12GB VRAM)
SDXL-DMD2 [55]	1018.5	
SDXL-Lightning [20]	1002.3	
SDXL-HyperSD [36]	990.5	
SANA-Sprint 1.6B [5]	932.1	
SD3.5-Flash 6-bit	**1090.2**	Mobile Phones (< 8GB VRAM)
SDv2.1 [37]	909.8	

DMD2 [55] and NitroFusion [3] (distilled from SDXL-DMD2) generate better texture but similarly perform worse in composition and result in artifacts (second row, cat on the book and first row, three owls). Comparatively, our method consistently generates high quality images and outperforms other 4-step pipelines in generation fidelity considerably. We also compare with 2-step pipelines like SANA-Sprint 1.6B [5]. SANA-Sprint [5] generates more details but with inconsistent style, sometimes generating stylistic images (first and third column) without style prompt. SANA-Sprint [53] also generates smudged facial features in non close-up environments (see fourth row). We also provide examples of our 4-step 16-bit model with and without T5 in Fig. 5.

5.4 Quantitative Comparisons

We conduct extensive quantitative validation (in Tab. 3) by generating 30K samples for captions from the COCO dataset [21], where we use metrics like **ImageReward** [54] **CLIPScore** [34], **FID** [12], and **Aesthetic Score** [43] to quantify generation performance. ImageReward (IR) and Aesthetic Score (AeS) are human preference metrics and are trained to reflect human preferences on image quality. Metrics like CLIPScore and FID are computed for quantifying text alignment and similarity to real images respectively. CLIPScore is measured as the similarity between text prompts and generated images in CLIP ViT-B/32 [18] semantic space. FID [12] is calculated as the distance between distributions of generated and real images (from COCO here) in the Inception-V3 [50] feature space. We also include comparisons on the **GenEval** [10] score where images of specific objects are generated in different settings and evaluated with an object detection framework for identifying text-to-image alignment. We compare against all baselines and competitors with these metrics along with their corresponding **Latency** (Tab. 4) as the time taken to generate a sample on a RTX 4090 GPU with 16-bit float precision (BF16) unless otherwise specified. From Tab. 3, we find that our method offers competitive performance for text to image generation compared to recent works like SDXL-DMD2 and NitroFusion, while surpassing the teacher model SD3.5M in metrics like GenEval, AeS and IR.

Discussion on FID: While FID increases after distilling SD3.5M, this change is not reflected as a semantic gap in other automated metrics or large-scale

user studies. Importantly, metrics like IR, AeS, and CLIPScore (Tab. 3) are computed on the same COCO dataset, but without significant degradation. Despite higher FID, we maintain strong semantic alignment, achieve SOTA AeS, and **outperform all competitors** in blind user studies. Notably, FID penalizes the superior SD3.5M (20.06) vs SDXL (14.72), highlighting its sensitivity to low-level shifts (e.g., sharpness) rather than perceptual quality. We hence attribute our drop in FID to its reliance on low-level image statistics and Inception-based features that do not necessarily correspond to semantic or perceptual quality. In our analysis, we prioritise human-preference from user studies over FID.

Table 3: Quantitative comparison: Comparison with other models on automated metrics. Models using SD3.5M or SD3M are coloured in green.

Methods	Steps	CLIP (↑)	AeS (↑)	IR (↑)	GenEval (↑)	FID (↓)
SDXL [32]	50	31.65	6.32	0.72	0.54	14.72
SD3.5M [48]	50	32.00	5.99	0.91	0.64	20.06
SDXL-Turbo [41]	4	31.67	6.19	0.84	0.56	20.76
SDXL-Lightning [20]	4	31.25	6.48	0.74	0.54	21.48
SDXL-DMD2 [55]	4	31.64	6.28	0.88	0.56	16.64
SDXL-HyperSD [36]	4	31.59	6.67	1.05	0.56	24.01
NitroFusion (Real.) [3]	4	31.28	6.41	0.91	0.55	22.66
HyperSD-SD3 [36]	8	31.39	6.18	0.95	0.65	23.95
SiD-DiT [59]	4	31.86	6.24	1.01	0.66	22.79
SD3.5M-DMD2 [55]	4	31.32	6.29	0.95	0.58	25.93
SWD-M [3]	4	32.00	6.37	1.12	0.72	25.90
SD3.5M-Turbo (w CFG) [51]	4	31.16	5.86	0.30	0.54	26.14
SANA-Sprint 0.6B [5]	2	31.39	6.54	0.98	0.77	24.99
SANA-Sprint 1.6B [5]	2	31.43	6.61	1.01	0.73	23.10
SD3.5-Flash 16-bit (+T5)	4	31.65	6.38	1.10	0.70	29.80
SD3.5-Flash 16-bit (No T5)	4	31.63	6.39	1.08	0.68	28.65
SD3.5-Flash 8-bit (+8-bit T5)	4	31.64	6.37	1.10	0.70	29.99
SD3.5-Flash 8-bit (No T5)	4	31.62	6.39	1.08	0.68	28.84

Table 4: Latency comparison of few-step generation pipelines. Models using SD3.5M or SD3M are coloured in green.

Methods	Steps	Latency (s) (↓)
SDXL-based 4-step [3, 20, 36, 41, 55]	4	0.43
SD3.5M-based 4-step [49, 59]	4	0.66
SD3.5M-Turbo (w CFG) [51]	4	1.06
SANA-Sprint 0.6B [5]	2	0.22
SANA-Sprint 1.6B [5]	2	0.24
SD3.5-Flash 16-bit (+T5)	4	0.58
SD3.5-Flash 16-bit (No T5)	4	0.55
SD3.5-Flash 8-bit (+8-bit T5)	4	0.66
SD3.5-Flash 8-bit (No T5)	4	0.61

5.5 Ablative Studies

We conduct ablative experiments (Fig. 7) specifically for the second stage of training by distilling SD3.5M (16-bit 4-step) without individual components in our pipeline, showing their importance for generation fidelity. Particularly, we distill the model: (i) **w/o Adversarial Objective**: where we do not use GAN training for guiding generation, (ii) **w/o Pre-Training**: Where we do not pre-train the student generator G_θ, (iii) **w/o Timestep Sharing**: Where we use random timestep t for x_t in $\mathcal{L}_{\text{DMD}}$ instead of those on the student trajectory, (iv) **w/o**

Discriminator Refresh: Where the discriminator heads are not periodically re-initialised to correct overfitting and (v) **w/o Split-Timestep Fine-tuning**: Where we do not train the final stage of the pipeline, included as a baseline. Some of these ablations include others, specifically training without adversarial objective includes discriminator refresh and in none of the cases we train the networks for the final stage (*i.e.* Split-Timestep fine-tuning). We train the ablation students for the same iterations as our student model's second stage of training and start from the same pre-trained checkpoint. We find that removing the adversarial objective destabilises training completely resulting in poor generation quality. Without pre-training, colour and composition are impacted the most with oversaturated samples. Training without timestep sharing also results in poor texture, colour, and composition leading to loss of severe details. Finally, without discriminator refresh we find minor compositional errors and over smooth images with low detail. Comparing the model before and after split-timestep fine-tuning, we observe increased details in background and better shadows along with superior composition. We also include a table comprising of quantitative analysis of these ablative studies in Tab. 5 using metrics defined in Sec. 5.4.

Fig. 7: Qualitative Ablation study: Demonstrating the importance of each component in our training pipeline.

Table 5: Quantitative Ablation Study: Quantitative analysis for ablative studies.

Ablation	AeS (↑)	CLIP (↑)	FID (↓)	IR (↑)
w/o Adversarial Objective	4.72	28.50	97.40	-0.57
w/o Pre-Training	6.22	31.14	30.13	0.89
w/o Timestep Sharing	5.97	30.97	35.61	0.67
w/o Discriminator Refresh	6.07	31.22	41.84	0.52
Baseline (w/o Split-Timestep Fine-tuning)	6.32	31.46	30.76	0.93
Ours (SD3.5-Flash 16-bit + T5)	**6.38**	**31.65**	**29.80**	**1.10**

Ablative user Study: To better demonstrate complex prompt understanding, we additionally ablate Split-Timestep Fine-tuning as an essential part of our

training pipeline. Using the same user study test set from Sec. 5.2, we compare generations from our final model against those produced before Split-Timestep Fine-tuning in an ablative user study. Results in Fig. 8 show that removing this stage leads to a significant drop in prompt alignment in complex prompts.

Fig. 8: Ablative User Study: Performance drops without Split-Timestep Fine-tuning

5.6 Generalisation Potential

We attempt to re-train existing baselines with public code on SD3.5M but obtain poor results with SANA TrigFlow and DMD based approaches (see Fig. 9-Top). We also train our pipeline (till stage 2) on larger models like FLUX-1.dev (∼6× parameters) and obtain reasonable samples with minor artifacts (see Fig. 9-Bottom). Our objective incorporates DMD2, which, as discussed in SenseFlow [9], requires modifications for scaling. That said, our primary goal is high-performance image generation on mobile devices, hence we focus on compressing SD3.5M via distillation, text encoder dropout, and quantization, to get models deployable on laptops and mobile phones. When compared with other methods like SiD-DiT [59] that scale to FLUX.1-dev, we find better performance on corresponding SD3.5M models: 45.6% preference for SiD-DiT compared to 54.4% for ours in Fig. 4.

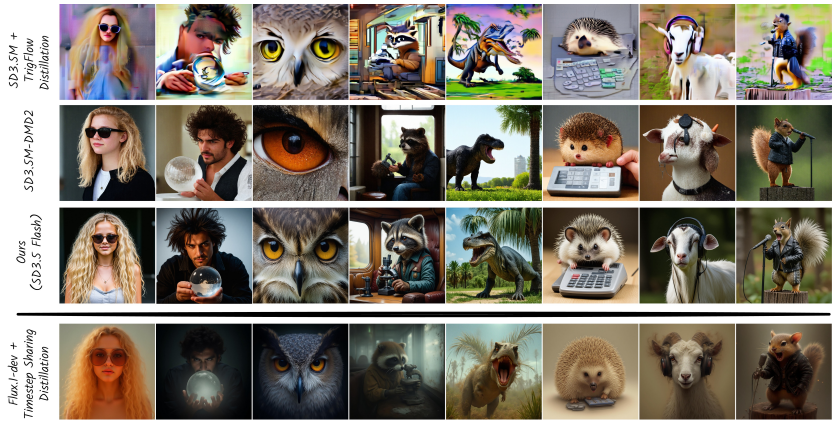

Fig. 9: Generalisation Potential: (i) Top - Applying competitor algorithms to SD3.5M to check for generalisation potential. (ii) Bottom - Applying our Timestep Sharing distillation paradigm to larger models like Flux.1-dev [1]

6 Conclusion

As in all distillation processes, we trade-off some aspect of quality and diversity with inference speed in complex generation tasks. We find that removing T5 for faster inference with lower memory also makes it difficult to construct complex compositions from worse conditional context (Fig. 5). However, these limitations are not unique to our method and are a natural consequence of approximating diffusion trajectories with low-step models. Despite them, we find our 4-step

model offers up-to $\sim 18\times$ speed-up on the teacher and surpasses it in average performance on large scale user studies with various levels of prompt complexity.

References

1. Black Forest Labs: Flux. `https://github.com/black-forest-labs/flux` (2024)
2. Bohan, O.B.: Taesd: Tiny autoencoder stable diffusion. `https://github.com/madebyollin/taesd` (2024)
3. Chen, D.Y., Bandyopadhyay, H., Zou, K., Song, Y.Z.: Nitrofusion: High-fidelity single-step diffusion through dynamic adversarial training. In: CVPR (2024)
4. Chen, J., Wu, Y., Luo, S., Xie, E., Paul, S., Luo, P., Zhao, H., Li, Z.: Pixart-{\delta}: Fast and controllable image generation with latent consistency models. arXiv preprint arXiv:2401.05252 (2024)
5. Chen, J., Xue, S., Zhao, Y., Yu, J., Paul, S., Chen, J., Cai, H., Xie, E., Han, S.: Sana-sprint: One-step diffusion with continuous-time consistency distillation. arXiv preprint arXiv:2503.09641 (2025)
6. Choi, J., Kim, M., Ahn, D., Kim, T., Kim, Y., Jo, D., Jeon, H., Kim, J.J., Kim, H.: Squeezing large-scale diffusion models for mobile. arXiv preprint arXiv:2307.01193 (2023)
7. Dao, T., Nguyen, T.H., Le, T., Vu, D., Nguyen, K., Pham, C., Tran, A.: Swiftbrush v2: Make your one-step diffusion model better than its teacher. In: ECCV (2024)
8. Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., et al.: Scaling rectified flow transformers for high-resolution image synthesis. In: ICML (2024)
9. Ge, X., Zhang, X., Xu, T., Zhang, Y., Zhang, X., Wang, Y., Zhang, J.: Senseflow: Scaling distribution matching for flow-based text-to-image distillation. arXiv preprint arXiv:2506.00523 (2025)
10. Ghosh, D., Hajishirzi, H., Schmidt, L.: Geneval: An object-focused framework for evaluating text-to-image alignment. In: NeurIPS (2023)
11. Ham, S., Woo, S., Kim, J.Y., Go, H., Park, B., Kim, C.: Diffusion model patching via mixture-of-prompts. In: AAAI (2025)
12. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: NeurIPS (2017)
13. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: NeurIPS (2020)
14. Karras, T., Aittala, M., Aila, T., Laine, S.: Elucidating the design space of diffusion-based generative models. In: NeurIPS (2022)
15. Kim, D., Lai, C.H., Liao, W.H., Murata, N., Takida, Y., Uesaka, T., He, Y., Mitsufuji, Y., Ermon, S.: Consistency trajectory models: Learning probability flow ode trajectory of diffusion. arXiv preprint arXiv:2310.02279 (2023)
16. Kingma, D.P., Welling, M., et al.: Auto-encoding variational bayes (2013)
17. Kohler, J., Pumarola, A., Schönfeld, E., Sanakoyeu, A., Sumbaly, R., Vajda, P., Thabet, A.: Imagine flash: Accelerating emu diffusion models with backward distillation. arXiv preprint arXiv:2405.05224 (2024)
18. Kolesnikov, A., Dosovitskiy, A., Weissenborn, D., Heigold, G., Uszkoreit, J., Beyer, L., Minderer, M., Dehghani, M., Houlsby, N., Gelly, S., Unterthiner, T., Zhai, X.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021)

19. Li, Y., Wang, H., Jin, Q., Hu, J., Chemerys, P., Fu, Y., Wang, Y., Tulyakov, S., Ren, J.: Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. In: NeurIPS (2023)
20. Lin, S., Wang, A., Yang, X.: Sdxl-lightning: Progressive adversarial diffusion distillation. arXiv preprint arXiv:2402.13929 (2024)
21. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014)
22. Lipman, Y., Chen, R.T., Ben-Hamu, H., Nickel, M., Le, M.: Flow matching for generative modeling. arXiv preprint arXiv:2210.02747 (2022)
23. Liu, S., Yu, W., Tan, Z., Wang, X.: Linfusion: 1 gpu, 1 minute, 16k image. arXiv preprint arXiv:2409.02097 (2024)
24. Liu, X., Gong, C., Liu, Q.: Flow straight and fast: Learning to generate and transfer data with rectified flow. arXiv preprint arXiv:2209.03003 (2022)
25. Liu, X., Zhang, X., Ma, J., Peng, J., et al.: Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In: ICLR (2023)
26. Lu, C., Song, Y.: Simplifying, stabilizing and scaling continuous-time consistency models. arXiv preprint arXiv:2410.11081 (2024)
27. Luhman, E., Luhman, T.: Knowledge distillation in iterative generative models for improved sampling speed. arXiv preprint arXiv:2101.02388 (2021)
28. Ma, Q., Ning, X., Liu, D., Niu, L., Zhang, L.: Decouple-then-merge: Towards better training for diffusion models. In: CVPR (2025)
29. Meng, C., Rombach, R., Gao, R., Kingma, D., Ermon, S., Ho, J., Salimans, T.: On distillation of guided diffusion models. In: CVPR (2023)
30. Nguyen, T.H., Tran, A.: Swiftbrush: One-step text-to-image diffusion model with variational score distillation. In: CVPR (2024)
31. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. TMLR (2023)
32. Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., Rombach, R.: Sdxl: Improving latent diffusion models for high-resolution image synthesis. arXiv preprint arXiv:2307.01952 (2023)
33. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. arXiv preprint arXiv:2209.14988 (2022)
34. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: ICML (2021)
35. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR (2020)
36. Ren, Y., Xia, X., Lu, Y., Zhang, J., Wu, J., Xie, P., Wang, X., Xiao, X.: Hyper-sd: Trajectory segmented consistency model for efficient image synthesis. arXiv preprint arXiv:2404.13686 (2024)
37. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: CVPR (2022)
38. Salimans, T., Ho, J.: Progressive distillation for fast sampling of diffusion models. arXiv preprint arXiv:2202.00512 (2022)
39. Sauer, A., Boesel, F., Dockhorn, T., Blattmann, A., Esser, P., Rombach, R.: Fast high-resolution image synthesis with latent adversarial diffusion distillation. In: SIGGRAPH Asia (2024)
40. Sauer, A., Karras, T., Laine, S., Geiger, A., Aila, T.: Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis. In: ICML (2023)

41. Sauer, A., Lorenz, D., Blattmann, A., Rombach, R.: Adversarial diffusion distillation. In: ECCV (2024)
42. Sauer, A., Schwarz, K., Geiger, A.: Stylegan-xl: Scaling stylegan to large diverse datasets. In: ACM SIGGRAPH (2022)
43. Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al.: Laion-5b: An open large-scale dataset for training next generation image-text models. In: NeurIPS (2022)
44. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502 (2020)
45. Song, Y., Dhariwal, P.: Improved techniques for training consistency models. arXiv preprint arXiv:2310.14189 (2023)
46. Song, Y., Dhariwal, P., Chen, M., Sutskever, I.: Consistency models. In: ICML (2023)
47. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456 (2020)
48. Stability AI: Sd3.5. `https://github.com/Stability-AI/sd3.5` (2024)
49. Starodubcev, N., Kuznedelev, D., Babenko, A., Baranchuk, D.: Scale-wise distillation of diffusion models. arXiv preprint arXiv:2503.16397 (2025)
50. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: CVPR (2016)
51. TensorArt Studios: stable-diffusion-3.5-medium-turbo (2025)
52. Wang, Z., Lu, C., Wang, Y., Bao, F., Li, C., Su, H., Zhu, J.: Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In: NeurIPS (2023)
53. Xie, E., Chen, J., Chen, J., Cai, H., Tang, H., Lin, Y., Zhang, Z., Li, M., Zhu, L., Lu, Y., et al.: Sana: Efficient high-resolution image synthesis with linear diffusion transformers. In: ICLR (2025)
54. Xu, J., Liu, X., Wu, Y., Tong, Y., Li, Q., Ding, M., Tang, J., Dong, Y.: Imagereward: learning and evaluating human preferences for text-to-image generation. In: NeurIPS (2023)
55. Yin, T., Gharbi, M., Park, T., Zhang, R., Shechtman, E., Durand, F., Freeman, B.: Improved distribution matching distillation for fast image synthesis. In: NeurIPS (2024)
56. Yin, T., Gharbi, M., Zhang, R., Shechtman, E., Durand, F., Freeman, W.T., Park, T.: One-step diffusion with distribution matching distillation. In: CVPR (2024)
57. Yu, J., Xu, Y., Koh, J.Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B.K., et al.: Scaling autoregressive models for content-rich text-to-image generation. arXiv preprint arXiv:2206.10789 (2022)
58. Zhao, Y., Xu, Y., Xiao, Z., Jia, H., Hou, T.: Mobilediffusion: Instant text-to-image generation on mobile devices. In: ECCV (2024)
59. Zhou, M., Gu, Y., Zheng, H., Song, L., He, G., Zhang, Y., Hu, W., Yang, Y.: Score distillation of flow matching models. arXiv preprint arXiv:2509.25127 (2025)