T-STITCH: ACCELERATING SAMPLING IN PRE-TRAINED DIFFUSION MODELS WITH TRAJECTORY STITCHING

Zizheng Pan[1]* **Bohan Zhuang**[1]† **De-An Huang**[2] **Weili Nie**[2] **Zhiding Yu**[2]

Chaowei Xiao[2,3] **Jianfei Cai**[1] **Anima Anandkumar**[4]

[1]Monash University [2]NVIDIA [3]University of Wisconsin, Madison [4]Caltech

ABSTRACT

Sampling from diffusion probabilistic models (DPMs) is often expensive for high-quality image generation and typically requires many steps with a large model. In this paper, we introduce sampling Trajectory Stitching (**T-Stitch**), a simple yet efficient technique to improve the sampling efficiency with little or no generation degradation. Instead of solely using a large DPM for the entire sampling trajectory, T-Stitch first leverages a smaller DPM in the initial steps as a cheap drop-in replacement of the larger DPM and switches to the larger DPM at a later stage. Our key insight is that different diffusion models learn similar encodings under the same training data distribution and smaller models are capable of generating good global structures in the early steps. Extensive experiments demonstrate that T-Stitch is training-free, generally applicable for different architectures, and complements most existing fast sampling techniques with flexible speed and quality trade-offs. On DiT-XL, for example, 40% of the early timesteps can be safely replaced with a 10x faster DiT-S without performance drop on class-conditional ImageNet generation. We further show that our method can also be used as a drop-in technique to not only accelerate the popular pretrained stable diffusion (SD) models but also improve the prompt alignment of stylized SD models from the public model zoo. Finally, the explicit model allocation strategy of T-Stitch significantly reduces the need of training or searching, delivering high deployment efficiency.

1 INTRODUCTION

Diffusion probabilistic models (DPMs) (Ho et al., 2020) have demonstrated remarkable success in generating high-quality data among various real-world applications, such as text-to-image generation (Rombach et al., 2022), audio synthesis (Kong et al., 2021) and 3D generation (Poole et al., 2023), etc. Achieving high generation quality, however, is expensive due to the need to sample from a large DPM, typically involving hundreds of denoising steps, each of which requires a high computational cost. For example, even with a high-performance RTX 3090, generating 8 images with DiT-XL (Peebles & Xie, 2022) takes 16.5 seconds with 100 denoising steps, which is $\sim 10\times$ slower than its smaller counterpart DiT-S (1.7s) with a lower generation quality.

Recent works tackle the inference efficiency issue by speeding up the sampling of DPMs in two ways: (1) reducing the computational costs per step or (2) reducing the number of sampling steps. The former approach can be done by model compression through quantization (Li et al., 2023b) and pruning (Fang et al., 2023), or by redesigning lightweight model architectures (Yang et al., 2023; Lee et al., 2023). The second approach reduces the number of steps either by distilling multiple denoising steps into fewer ones (Salimans & Ho, 2022; Song et al., 2023; Zheng et al., 2023; Luo

*Work done during internship at NVIDIA

†Corresponding author: bohan.zhuang@gmail.com

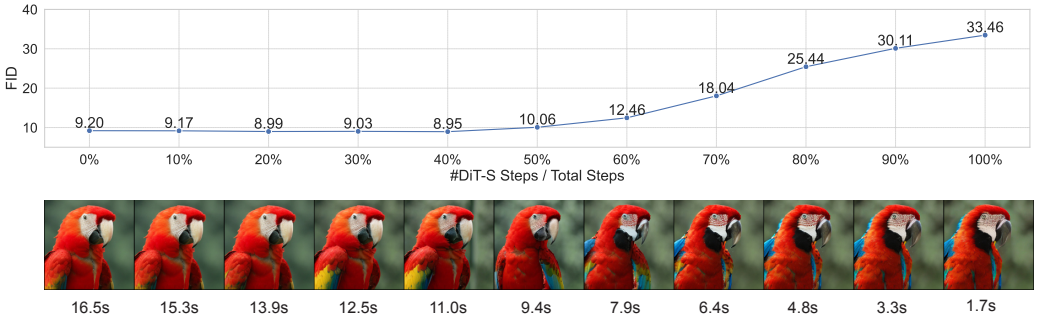


Figure 1: **Top:** FID comparison on class-conditional ImageNet when progressively stitching more DiT-S steps at the beginning and fewer DiT-XL steps in the end, based on DDIM 100 timesteps and a classifier-free guidance scale of 1.5. FID is calculated by sampling 5000 images. **Bottom:** One example of stitching more DiT-S steps to achieve faster sampling, where the time cost is measured by generating 8 images on one RTX 3090 in seconds (s).

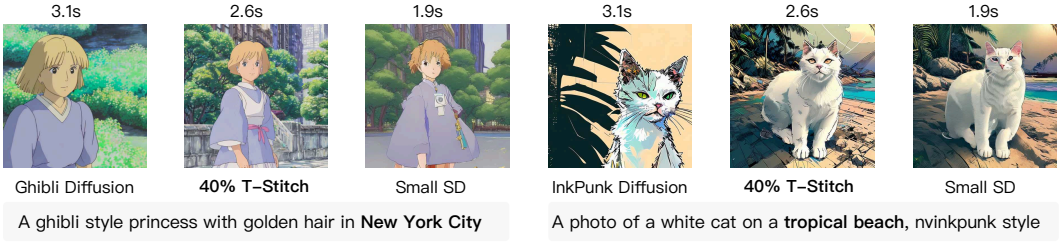


Figure 2: By directly adopting a small SD in the model zoo, T-Stitch naturally interpolates the speed, style, and image contents with a large styled SD, which also potentially improves the prompt alignment, *e.g.*, "New York City" and "tropical beach" in the above examples.

et al., 2023; Sauer et al., 2023) or by improving the differential equation solver (Song et al., 2021a; Lu et al., 2022; Zheng et al., 2023). While both directions can improve the efficiency of large DPMs, they assume that the computational cost of each denoising step remains the same, and a single model is used throughout the process. *However, we observe that different steps in the denoising process exhibit quite distinct characteristics, and using the same model throughout is a suboptimal strategy for efficiency.*

Our Approach. In this work, we propose *Trajectory Stitching* (**T-Stitch**), a simple yet effective strategy to improve DPMs' efficiency that complements existing efficient sampling methods by dynamically allocating computation to different denoising steps. *Our core idea is to apply small DPMs at the early denoising steps instead of using the same large model at all steps, as in previous works.* We show that by first applying a smaller DPM in the early denoising steps followed by switching to a larger DPM in the later denoising steps, we can reduce the overall computational costs *without* sacrificing the generation quality. Figure 1 shows an example of our approach using two DiT models (DiT-S and DiT-XL), where DiT-S is computationally much cheaper than DiT-XL. With the increase in the percentage of steps from DiT-S instead of DiT-XL in our T-stitch, we can keep increasing the inference speed. In our experiments, we find that there is no degradation of the generation quality (in FID), even when the first 40% of steps are using DiT-S, leading to around 1.5$\times$ *lossless* speedup.

Some prior arts Liu et al. (2023); Yang et al. (2024) share a similar spirit with ours, i.e., applying different models at different steps. However, compared to their costly trained and searched model schedules, our method is distinct in two new key insights: (1) Recent work suggests a common latent space across different DPMs trained on the same data distribution (Song et al., 2021b; Roeder et al., 2021). Thus, different DPMs tend to share similar sampling trajectories, which makes it possible to stitch across different model sizes and even architectures, *without significant training or searching*. (2) From the frequency perspective, the denoising process focuses on generating low-frequency components at the early steps while the later steps target the high-frequency signals (Yang et al., 2023). Although the small models are not as effective for high-frequency details, they can still generate a good global structure at the beginning. In our experiments, we show that T-Stitch achieves comparable or even better trade-offs than training/searching-based model schedules.

With comprehensive experiments, we demonstrate that T-Stitch substantially speeds up large DPMs without much loss of generation quality. This observation is consistent across a spectrum of architectures and diffusion model samplers. This also implies that T-Stitch can be directly applied to widely used large DPMs without any re-training (*e.g.*, Stable Diffusion (SD) Rombach et al. (2022)). Figure 2 shows the results of speeding up stylized Stable Diffusion with a relatively smaller pretrained SD model Kim et al. (2023). Surprisingly, we find that T-Stitch not only improves speed but also *improves prompt alignment* for stylized models. This is possibly because the fine-tuning process of stylized models (*e.g.*, ghibli, inkpunk) degrades their prompt alignment. T-Stitch improves both efficiency and generation quality here by combining small SD models to complement the prompt alignment for large SD models specialized in stylizing the image.

Note that T-Stitch is *complementary* to most existing fast sampling approaches. The part of the trajectory that is taken by the large DPM can still be sped up by reducing the number of steps taken by it, or by reducing its computational cost with compression techniques. In addition, while T-Stitch can already effectively improve the quality-efficiency trade-offs without any overhead of re-training, we show that the generation quality of T-Stitch can be further improved when we fine-tune the stitched DPMs given a trajectory schedule (Section A.13). By fine-tuning the large DPM only on the timesteps that it is applied, the large DPM can better specialize in providing high-frequency details and further improve generation quality. Furthermore, we show that the training-free Pareto frontier generated by T-Stitch improves quality-efficiency trade-offs to training-based methods designed for interpolating between neural network models via model stitching (Pan et al., 2023a;b). Note that T-Stitch is not limited to only two model sizes, and is also applicable to different DPM architectures.

We summarize our main contributions as follows:

- We propose T-Stitch, a simple yet highly effective approach for improving the inference speed of DPMs, by applying a small DPM at early denoising steps while a large DPM at later steps. Without retraining, we achieve better speed and quality trade-offs than individual large DPMs and even non-trivial lossless speedups.
- We conduct extensive experiments to demonstrate that our method is generally applicable to different model architectures and samplers, and is complementary to existing fast sampling techniques.
- Notably, without any re-training overhead, T-Stitch not only accelerates Stable Diffusion models that are widely used in practical applications but also improves the prompt alignment of stylized SD models for text-to-image generation.

2 Related Works

Efficient diffusion models. Despite the success, DPMs suffer from the slow sampling speed (Rombach et al., 2022; Ho et al., 2020) due to hundreds of timesteps and the large denoiser (*e.g.*, U-Net). To expedite the sampling process, some efforts have been made by directly utilizing network compression techniques to diffusion models, such as pruning (Fang et al., 2023) and quantization (Shang et al., 2023; Li et al., 2023b). On the other hand, many works seek for reducing sampling steps, which can be achieved by distillation (Salimans & Ho, 2022; Zheng et al., 2023; Song et al., 2023; Luo et al., 2023; Sauer et al., 2023), implicit sampler (Song et al., 2021a), and improved differential equation (DE) solvers (Lu et al., 2022; Song et al., 2021b; Jolicoeur-Martineau et al., 2021; Liu et al., 2022). Another line of work also considers accelerating sampling by parallel sampling Zheng et al. (2023); Shih et al. (2023). As a complementary technique to the above methods, T-Stitch accelerates large DPM sampling by leveraging pretrained small DPMs at early denoising steps, while leaving sufficient space for large DPMs at later steps. Some recent works also explore using different models at different sizes to achieve better speed-quality trade-offs. For example, OMS-DPM Liu et al. (2023) trains a predictor that predicts the model performance given a model schedule. DDSM Yang et al. (2024) trains a supernet and adopts evolutionary search. Compared to them, T-Stitch is training-free. Under a time budget, the search in T-Stitch is much more efficient since the fraction of the small DPM directly defines a unique efficiency and generation quality trade-off.

Multiple experts in diffusion models. Previous observations have revealed that the synthesis behavior in DPMs can change at different timesteps (Balaji et al., 2022; Yang et al., 2023), which has inspired some works to propose an ensemble of experts at different timesteps for better performance.

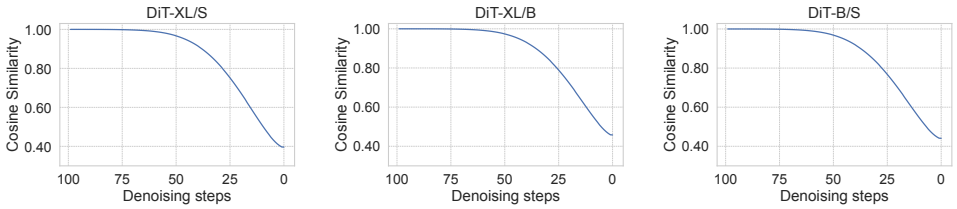


Figure 3: Similarity comparison of latent embeddings at different denoising steps between different DiT models. Results are averaged over 32 images.

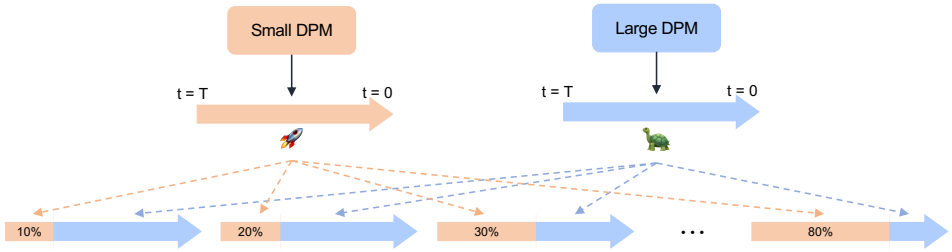


Figure 4: **Trajectory Stitching** (T-Stitch): Based on pretrained small and large DPMs, we can leverage the more efficient small DPM with different percentages at the early denoising sampling steps to achieve different speed-quality trade-offs.

For example, Balaji et al. (2022) trained an ensemble of expert denoisers at different denoising intervals. However, allocating multiple large denoisers linearly increases the model parameters and does not reduce the computational cost. Yang et al. (2023) proposed a lite latent diffusion model (*i.e.*, LDM) which incorporates a gating mechanism for the wavelet transform in the denoiser to control the frequency dynamics at different steps, which can be regarded as an ensemble of frequency experts. Following the same spirit, Lee et al. (2023) allocated different small denoisers at different denoising intervals to specialize on their respective frequency ranges. Nevertheless, most existing works adopt the same-sized model over all timesteps, which barely consider the speed and quality trade-offs between different-sized models. In contrast, we explore a flexible trade-off between small and large DPMs and reveal that the early denoising steps can be sufficiently handled by a much efficient small DPM.

Stitchable neural networks. Stitchable neural networks (SN-Net) (Pan et al., 2023a) is motivated by the idea of model stitching (Lenc & Vedaldi, 2015; Bansal et al., 2021; Csiszárik et al., 2021; Yang et al., 2022), where the pretrained models of different scales within a pretrained model family can be splitted and stitched together with simple stitching layers (*i.e.*, 1×1 convs) without a significant performance drop. Based on the insight, SN-Net inserts a few stitching layers among models of different sizes and applies joint training to obtain numerous networks (*i.e.*, stitches) with different speed-performance trade-offs. The following work of SN-Netv2 (Pan et al., 2023b) enlarges its space and demonstrates its effectiveness on downstream dense prediction tasks. In this work, we compare our technique with SN-Netv2 to show the advantage of trajectory stitching over model stitching in terms of the speed and quality trade-offs in DPMs. Our T-Stitch is a better, simpler and more general solution.

3 Trajectory Stitching

In this section, we introduce Trajectory Stitching (T-Stitch) with a focus on the class of score-based diffusion models in a continuous time (Song et al., 2021b). Due to the limited space, we leave the preliminary in Appendix Section A.1.

Why can different pretrained DPMs be directly stitched along the sampling trajectory? First of all, DPMs from the same model family usually takes the latent noise inputs and outputs of the same shape, (*e.g.*, $4 \times 32 \times 32$ in DiTs). There is no dimension mismatch when applying different DPMs at different denoising steps. More importantly, as pointed out in (Song et al., 2021b), different DPMs that are trained on the same dataset often learn similar latent embeddings. We observe that this is especially true for the latent noises at early denoising sampling steps, as shown in Figure 3, where the cosine similarities between the output latent noises from different DiT models reach almost 100% at

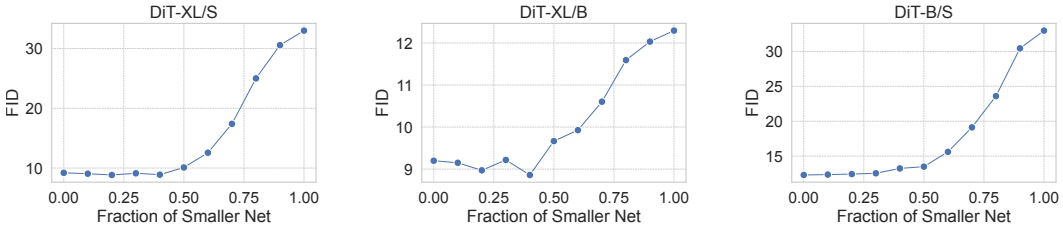


Figure 5: T-Stitch of two model combinations: DiT-XL/S, DiT-XL/B and DiT-B/S. We adopt DDIM 100 timesteps with a classifier-free guidance scale of 1.5.

early steps. This motivates us to propose T-Stitch, a novel step-level stitching strategy that leverages a pretrained small model at the beginning to accelerate the sampling speed of large diffusion models.

Principle of model selection. Figure 4 shows the framework of our proposed T-Stitch for different speed-quality tradeoffs. In principle, the fast speed or worst generation quality we can achieve is roughly bounded by the smallest model in the trajectory, whereas the slowest speed or best generation quality is determined by the largest denoiser. Thus, given a large diffusion model that we want to speed up, we select a small model that is 1) clearly faster, 2) sufficiently optimized, and 3) trained on the same dataset as the large model or at least they have learned similar data distributions (*e.g.*, pretrained or finetuned stable diffusion models).

Pairwise model allocation. By default, T-Stitch adopts a pairwise denoisers in the sampling trajectory as it performs very well in practice. Specifically, we first define a denoising interval as a range of sampling steps in the trajectory, and the fraction of it over the total number of steps T is denoted as r, where $r \in [0, 1]$. Next, we treat the model allocation as a compute budget allocation problem. From Figure 3, we observe that the latent similarity between different scaled denoisers keeps decreasing when T flows to 0. To this end, our allocation strategy adopts a small denoiser as a cheap replacement at the initial intervals then applies the large denoiser at the later intervals. In particular, suppose we have a small denoiser D_1 and a large denoiser D_2. Then we let D_1 take the first $\lfloor r_1 T \rceil$ steps and D_2 takes the last $\lfloor r_2 T \rceil$ steps, where $\lfloor \cdot \rceil$ denotes a rounding operation and $r_2 = 1 - r_1$. By increasing r_1, we naturally interpolate the compute budget between the small and large denoiser and thus obtain flexible quality and efficiency trade-offs. For example, in Figure 1, the configuration $r_1 = 0.5$ uniquely defines a trade-off where it achieves 10.06 FID and $1.76\times$ speedup.

More denoisers for more trade-offs. Note that T-Stitch is not limited to the pairwise setting. In fact, we can adopt more denoisers in the sampling trajectory to obtain more speed and quality trade-offs and a better Pareto frontier. For example, by using a medium sized denoiser in the intermediate interval, we can change the fractions of each denoiser to obtain more configurations. In practice, given a compute budget such as time cost, we can efficiently find a few configurations that satisfy this constraint via a pre-computed lookup table, as discussed in Section A.2.

Remark. Compared to existing multi-experts DPMs, T-Stitch directly applies models of *different sizes* in a *pretrained* model family. Thus, given a compute budget, we consider how to allocate different resources across different steps while benefiting from training-free. Furthermore, speculative decoding (Leviathan et al., 2023) shares a similar motivation with us, *i.e.*, leveraging a small model to speed up large language model sampling. However, this technique is specifically designed for autoregressive models, whereas it is not straightforward to apply the same sampling strategy to diffusion models. On the other hand, our method utilizes the DPM's property and achieves effective speedup.

4 Experiments

In this section, we first show the effectiveness of T-Stitch based on DiT (Peebles & Xie, 2022) as it provides a convenient model family. Then we extend into U-Net and Stable Diffusion models. Last, we ablate our technique with different sampling steps, and samplers to demonstrate that T-Stitch is generally applicable in many scenarios.

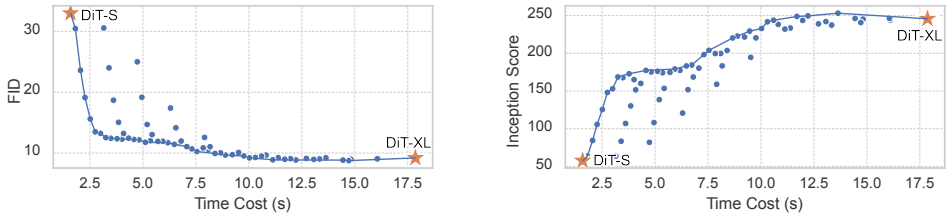


Figure 6: T-Stitch based on three models: DiT-S, DiT-B and DiT-XL. We adopt DDIM 100 timesteps with a classifier-free guidance scale of 1.5. We highlight the Pareto frontier in lines.

Table 1: T-Stitch with LDM (Rombach et al., 2022) and LDM-S on class-conditional ImageNet. All evaluations are based on DDIM and 100 timesteps. We adopt a classifier-free guidance scale of 3.0. The time cost is measured by generating 8 images on one RTX 3090.

Fraction of LDM-S	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
FID	20.11	19.54	18.74	18.64	18.60	19.33	21.81	26.03	30.41	35.24	40.92
Inception Score	199.24	201.87	202.81	204.01	193.62	175.62	140.69	110.81	90.24	70.91	54.41
Time Cost (s)	7.1	6.7	6.2	5.8	5.3	4.9	4.5	4.1	3.6	3.1	2.9

4.1 DiT Experiments

Implementation details. Following DiT, we conduct the class-conditional ImageNet experiments based on pretrained DiT-S/B/XL under 256×256 images and patch size of 2. A detailed comparison of the pretrained models is shown in Table 6. As T-Stitch is training-free, for two-model setting, we directly allocate the models into the sampling trajectory under our allocation strategy described in Section 3. For three-model setting, we enumerate all possible configuration sets by increasing the fraction by 0.1 per model one at a time, which eventually gives rise to 66 configurations that include pairwise combinations of DiT-S/XL, DiT-S/B, DiT-S/XL, and three model combinations DiT-S/B/XL. By default, we adopt a classifier-free guidance scale of 1.5 as it achieves the best FID for DiT-XL, which is also the target model in our setting.

Evaluation metrics. We adopt Fréchet Inception Distance (FID) (Heusel et al., 2017) as our default metric to measure the overall sample quality as it captures both diversity and fidelity (lower values indicate better results). Additionally, we report the Inception Score as it remains a solid performance measure on ImageNet, where the backbone Inception network (Szegedy et al., 2016) is pretrained. We use the reference batch from ADM (Dhariwal & Nichol, 2021) and sample 5,000 images to compute FID. In the supplementary material, we show that sampling more images (*e.g.*, 50K) does not affect our observation. By default, the time cost is measured by generating 8 images on a single RTX 3090 in seconds.

Results. Based on the pretrained model families, we first apply T-Stitch with any two-model combinations, including DiT-XL/S, DiT-XL/B, and DiT-B/S. For each setting, we begin the sampling steps with a relatively smaller model and then let the larger model deal with the last timesteps. In Figure 5, we report the FID comparisons on different combinations. In general, we observe that using a smaller model at the early 40-50% steps brings a minor performance drop for all combinations. Besides, the best/worst performance is roughly bounded by the smallest and largest models in the pretrained model family.

Furthermore, we show that T-Stitch can adopt a medium-sized model at the intermediate denoising intervals to achieve more speed and quality trade-offs. For example, built upon the three different-sized DiT models: DiT-S, DiT-B, DiT-XL, we start with DiT-S at the beginning then use DiT-B at the intermediate denoising intervals, and finally adopt DiT-XL to draw fine local details. Figure 6 indicates that the three-model combinations effectively obtain a smooth Pareto Frontier for both FID and Inception Score. In particular, at the time cost of ∼10s, we achieve 1.7× speedups with comparable FID (9.21 vs. 9.19) and Inception Score (243.82 vs. 245.73). We show the effect of using different classifier-free guidance scales in Section A.5.

4.2 U-Net Experiments

In this section, we show T-Stitch is complementary to the architectural choices of denoisers. We experiment with prevalent U-Net as it is widely adopted in many diffusion models. We adopt the

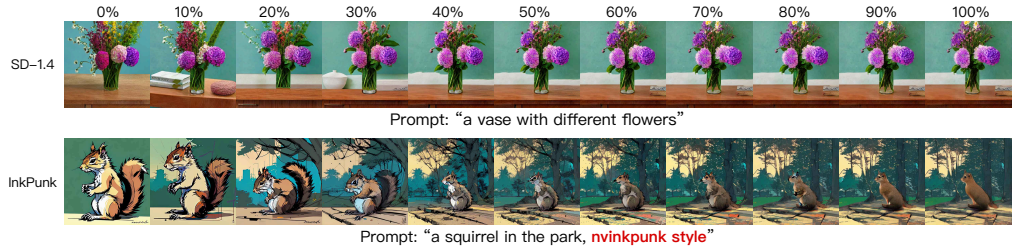


Figure 7: Based on a general pretrained small SD model, T-Stitch simultaneously accelerates a large general SD and complements the prompt alignment with image content when stitching other finetuned/stylized large SD models, *i.e.*, "park" in InkPunk Diffusion. Better viewed when zoomed in digitally.

Table 2: T-Stitch with BK-SDM Tiny (Kim et al., 2023) and SD v1.4. We report FID, Inception Score (IS) and CLIP score (Hessel et al., 2021) on MS-COCO 256×256 benchmark. The time cost is measured by generating one image on one RTX 3090.

Fraction of BK-SDM Tiny	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
FID	13.07	12.59	12.29	12.54	13.65	14.98	15.69	16.57	16.92	16.80	17.15
Inception Score	36.72	36.12	34.66	33.32	32.48	31.72	31.48	30.83	30.53	30.48	30.00
CLIP Score	0.2957	0.2957	0.2938	0.2910	0.2860	0.2805	0.2770	0.2718	0.2692	0.2682	0.2653
Time Cost (s)	3.1	3.0	2.9	2.8	2.6	2.5	2.4	2.3	2.1	2.0	1.9

class-conditional ImageNet implementation from the latent diffusion model (LDM) (Rombach et al., 2022). Based on their official implementation, we simply scale down the network channel width from 256 to 64 and the context dimension from 512 to 256. This modification produces a 15.8× smaller LDM-S. A detailed comparison between the two pretrained models is shown in Table 7.

Results. We report the results on T-Stitch with U-Net in Table 1. In general, under DDIM and 100 timesteps, we found the first ∼50% steps can be taken by an efficient LDM-S with comparable or even better FID and Inception Scores. At the same time, we observe an approximately linear decrease in time cost when progressively using more LDM-S steps in the trajectory. Overall, the U-Net experiment indicates that our method is applicable to different denoiser architectures. We provide the generated image examples in Section A.15 and also show that T-Stitch can be applied with even different model families in Section A.11.

4.3 Text-to-Image Stable Diffusion

Benefiting from the public model zoo on Diffusers (von Platen et al., 2022), we can directly adopt a small SD model to accelerate the sampling speed of any large pretrained or styled SD models without any training. In this section, we show how to apply T-Stitch to accelerate existing SD models in the model zoo. Previous research from Kim et al. (2023) has produced multiple SD models with different sizes by pruning the original SD v1.4 and then applying knowledge distillation. We then directly adopt the smallest model BK-SDM Tiny for our stable diffusion experiments. By default, we use a guidance scale of 7.5 under 50 steps using PNDM (Liu et al., 2022) sampler.

Results. In Table 2, we report the results by applying T-Stitch to the original SD v1.4. In addition to the FID and Inception Score, we also report the CLIP score for measuring the alignment of the image with the text prompt. Overall, we found the early 30% steps can be taken by BK-SDM Tiny without a significant performance drop in Inception Score and CLIP Scores while achieving even better FID. We believe a better and faster small model in future works can help to achieve better quality and efficiency trade-offs. Furthermore, we demonstrate that T-Stitch is compatible with other large SD models. For example, as shown in Figure 7, under the original SD v1.4, we achieve a promising speedup while obtaining comparable image quality. Moreover, with other stylized SD models such as Inkpunk style[1], we observe a natural style interpolation between the two models. More importantly, by adopting a small fraction of steps from a general small SD, we found it helps the image to be more aligned with the prompt, such as the "park" in InkPunk Diffusion. In this case,

[1]https://huggingface.co/Envvi/Inkpunk-Diffusion

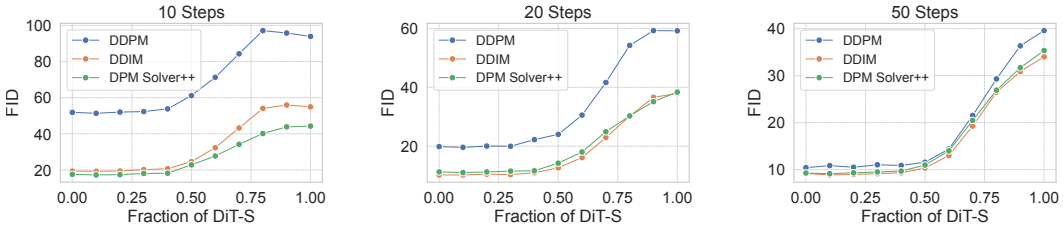


Figure 8: Effect of T-Stitch with different samplers, under guidance scale of 1.5.

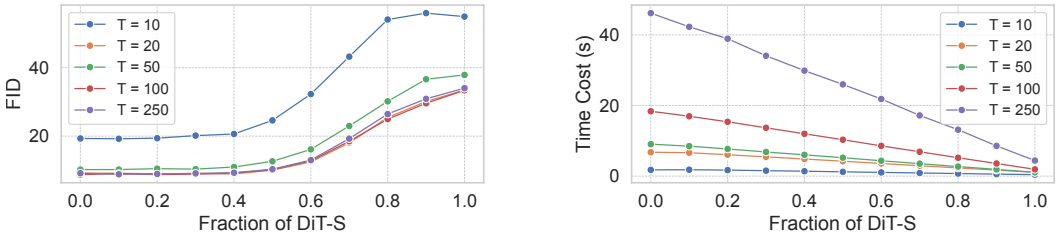


Figure 9: **Left:** We compare FID between different numbers of steps. **Right:** We visualize the time cost of generating 8 images under different number of steps, based on DDIM and a classifier-guidance scale of 1.5. "T" denotes the number of sampling steps.

we assume finetuning in these stylized SD may unexpectedly hurt prompt alignment, while adopting the knowledge from a general pretrained SD can complement the early global structure generation. Overall, this strongly supports another practical usage of T-Stitch: *Using a small general expert at the beginning for fast sketching and better prompt alignment, while letting any stylized SD at the later steps for patiently illustrating details.* Furthermore, we show that T-Stitch is compatible with ControlNet, SDXL, LCM in Section A.12 and zero-shot Text-to-Video generation in Section A.20.

4.4 Ablation Study

Effect of T-Stitch with different steps. To explore the efficiency gain on different numbers of sampling steps, we conduct experiments based on DDIM and DiT-S/XL. As shown in Figure 9, T-Stitch achieves consistent efficiency gain when using different number of steps and diffusion model samplers. In particular, we found the 40% early steps can be safely taken by DiT-S without a significant performance drop. It indicates that small DPMs can sufficiently handle the early denoising steps where they mainly generate the low-frequency components. Thus, we can leave the high-frequency generation of fine local details to a more capable DiT-XL. This is further evidenced by the generation examples in Figure 17, where we provide the sampled images at all fractions of DiT-S steps across different total number of steps. Overall, we demonstrate that T-Stitch is not competing but complementing other fast diffusion approaches that focus on reducing sampling steps.

Effect of T-Stitch with different samplers. Here we show T-Stitch is also compatible with advanced samplers (Lu et al., 2022) for achieving better generation quality in fewer timesteps. To this end, we experiment with prevalent samplers to demonstrate the effectiveness of T-Stitch with these orthogonal techniques: DDPM (Ho et al., 2020), DDIM (Song et al., 2021a) and DPM-Solver++ (Lu et al., 2022). In Figure 8, we use the DiT-S to progressively replace the early steps of DiT-XL under different samplers and steps. In general, we observe a consistent efficiency gain at the initial sampling stage, which strongly justifies that our method is a complementary solution with existing samplers for accelerating DPM sampling.

T-Stitch vs. model stitching. Previous works (Pan et al., 2023a;b) such as SN-Net have demonstrated the power of model stitching for obtaining numerous *architectures* that with different complexity and performance trade-offs. Thus, by adopting one of these architectures as the denoiser for sampling, SN-Net naturally achieves flexible quality and efficiency trade-offs. To show the advantage of T-Stitch in the Pareto frontier, we conduct experiments to compare with the framework of model stitching proposed in SN-Netv2 (Pan et al., 2023b). We provide implementation details in Section A.9. In Figure 10, we compare T-Stitch with model stitching based on DDIM sampler and 100 steps. Overall, while both methods can obtain flexible speed and quality trade-offs, T-Stitch achieves clearly better advantage over model stitching across different metrics.

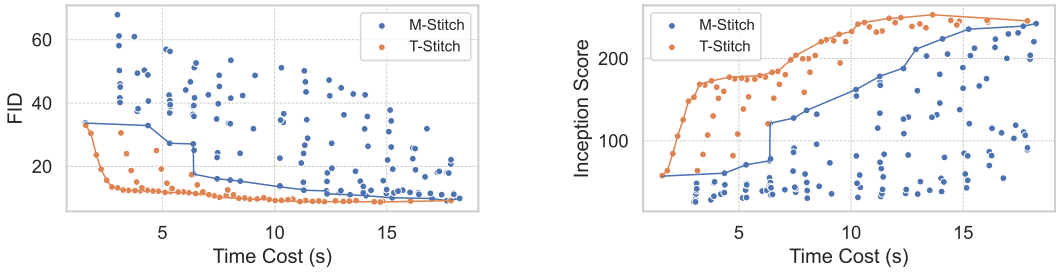


Figure 10: T-Stitch vs. model stitching (M-Stitch) Pan et al. (2023b) based on DiTs and DDIM 100 steps, with a classifier-free guidance scale of 1.5.

Table 3: Comparison between T-Stitch and training/searching-based model schedules on CIFAR-10. "#Eval Runs" indicates the number of FID evaluation runs required for both training and searching. Baselines are the largest networks in their model families. For T-Stitch, we only adopt the smallest network in their model family to accelerate the baselines.

(a) Comparison with DDSM Yang et al. (2024).

Name	Time per Img (s)	FID	#Eval Runs
Baseline	0.70	7.9	-
DDSM	0.45	8.3	500
T-Stitch	0.44	8.1	1

(b) Comparison with OMS-DPM Liu et al. (2023).

Name	Time per Img (s)	FID	#Eval Runs
Baseline	0.13	8.4	-
OMS-DPM	0.03	11.2	>6,000
T-Stitch	0.04	11.5	1

Compared to training-based acceleration methods. The widely adopted training-based methods for accelerating DPM sampling mainly include lightweight model design Zhao et al. (2023); Lee et al. (2023), model compression Kim et al. (2023), and steps distillation Salimans & Ho (2022); Song et al. (2023); Luo et al. (2023). Compared to them, T-Stitch is a training-free and *complementary acceleration technique* since it is agnostic to individual model optimization. In practice, T-Stitch achieves wide compatibility with different denoiser architectures (DiT and U-Net, Section 4.1 and Section 4.2), and any already pruned (Section A.8) or step-distilled models (Section A.17).

Compared to other training-free acceleration methods. Recent works Li et al. (2023a); Ma et al. (2023); Wimbauer et al. (2023) proposed to cache the intermediate feature maps in U-Net during sampling for speedup. T-Stitch is also complementary to these cache-based methods since the individual model can still be accelerated with caching, as shown in Section A.18. In addition, T-Stitch can also enjoy the benefit from model quantization Shang et al. (2023); Li et al. (2023b), VAE decoder acceleration Kodaira et al. (2023) and token merging Bolya et al. (2023) (Section A.19) since they are orthogonal approaches.

Compared to training/searching-based model schedules. Given a time budget such as the time cost for generating an image, there is a line of work Yang et al. (2024); Liu et al. (2023) searching for a model schedule. In contrast to these methods, we provide a clear allocation strategy, which essentially reduces the search space, thus significantly reducing the search cost, compared to training or searching-based methods which have a much larger search space. For example, under 1,000 steps and 4 different network widths in DDSM (Yang et al., 2024), the search space is $4^{1,000}$. Our work points out a simple but effective way by progressively allocating a small model at the early steps, resulting in only 1,000 configurations at its most fine-grained level. Benefiting from such a small space, our T-Stitch is practically way faster than existing works since the corresponding model schedule *under a time cost* can be obtained by directly querying a pre-calculated lookup table on CPU within a second. At the same time, our result is comparable or even better, as shown in Table 3.

Additional memory and storage overhead of T-Stitch. Intuitively, T-Stitch adopts a small DPM which can introduce additional memory and storage overhead. However, in practice, the large DPM is still the main bottleneck of memory and storage consumption. In this case, the additional overhead from small DPM is considerably minor. For example, as shown in Table 4, compared to DiT-XL, T-Stitch by adopting 50% steps of DiT-S only introduces additional 5% parameters, 4% GPU memory cost, 10% local storage cost, while significantly accelerating DiT-XL sampling speed by 1.76×.

Compared with more stitching baselines. By default, we design T-Stitch to start from a small DPM and then switch into a large DPM for the last denoising sampling steps. To show the effectiveness of this design, we compare our method with several baselines in Table 5 based on DiT-S and DiT-XL. These baselines are listed as below,

Table 4: Local storage and memory cost comparison between DiT-S, DiT-XL and T-Stitch. Memory and time cost are measured by generating 8 images in parallel on one RTX 3090.

Name	Parameter (M)	Local Storage (MB)	Memory Cost (MB)	Time Cost (s)
DiT-S	33	263	3,088	1.7
DiT-XL	675	2,576	3,166	16.5
T-Stitch (50%)	708 ($\times$1.04)	2,839 ($\times$1.10)	3,296 ($\times$1.04)	9.4 ($\times$1.76)

Table 5: Compared to other trajectory stitching baselines based on DiT-S/XL, DDIM 100 steps and guidance scale of 1.5. FID is calculated by 5K images. Memory and time cost are measured by a batch size of 8 on one RTX 3090.

Method	FID	Inception Score	Time Cost
Interleave	19.02	120.04	10.1
Decreasing Prob	12.94	163.45	9.8
Large to Small	27.61	72.60	10.0
Small to Large (Ours)	10.06	200.81	9.9

- **Interleaving.** During denoising sampling, we interleave the small and large model along the trajectory. Eventually, DiT-S takes 50% steps and DiT-XL takes another 50% steps.
- **Decreasing Prob.** Linearly decreasing the probability of using DiT-S from 1 to 0 during the denoising sampling steps.
- **Large to Small.** Adopting the large model at the early 50% steps and the small model at the last 50% steps.
- **Small to Large (our default design).** The default strategy of T-Stitch by adopting DiT-S at the early 50% steps and using DiT-XL at the last 50% steps.

As shown in Table 5, in general, our default design achieves the best FID and Inception Score with similar sampling speed, which strongly demonstrate its effectiveness.

Remarks on the small model. While a small model is required for T-Stitch, our key motivation is that the increasingly expanded large public model zoo more or less addressed this concern in practice. First, under T-Stitch, a small model that lies in a similar data distribution as the target large model can accelerate all its finetuned or stylized versions for free. For example, a single BK-SDM Tiny is able to simultaneously accelerate SDv1.4, or any stylized SDs based on it, *e.g.*, ControlNet, as shown in Section A.12. Furthermore, T-Stitch also works for different model families. Benefiting from it, we can easily accelerate U-ViT H sampling with a small DiT-S, as shown in Table 8. Second, free lunch is generally available in large public model zoos. The open-source community (*e.g.*, HuggingFace.co and Civitai.com) has released tens of thousands of pretrained models, including different scales (Segmind, 2023; Kim et al., 2023), which are pruned and knowledge distilled versions of large stable diffusion models. In our experiments, we have never trained a SD model from scratch, while being able to demonstrate broad applications of T-Stitch on SDXL, ControlNet, and step-distilled diffusion models.

5 Conclusion

We have proposed Trajectory Stitching, an effective and general approach to accelerate existing pretrained large diffusion model sampling by directly leveraging pretrained smaller counterparts at the initial denoising process, which achieves better speed and quality trade-offs than using an individual large DPM. Comprehensive experiments have demonstrated that T-Stitch achieved consistent efficiency gain across different model architectures, samplers, as well as various stable diffusion models. Besides, our work has revealed the power of small DPMs at the early denoising process. Future work may consider disentangling the sampling trajectory by redesigning or training experts of different sizes at different denoising intervals. For example, designing a better, faster small DPM at the beginning to draw global structures, then specifically optimizing the large DPM at the later stages to refine image details. Besides, more guidelines for the optimal trade-off and more in-depth analysis of the prompt alignment for stylized SDs can be helpful, which we leave for future work.

REFERENCES

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *CoRR*, abs/2211.01324, 2022.

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. In *NeurIPS*, pp. 225–236, 2021.

Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *CVPR*, 2023.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *ICLR*, 2023.

Adrián Csiszárik, Péter Korösi-Szabó, Ákos K. Matszangosz, Gergely Papp, and Dániel Varga. Similarity and matching of neural network representations. In *NeurIPS*, pp. 5656–5668, 2021.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 34:8780–8794, 2021.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. *arXiv preprint arXiv:2305.10924*, 2023.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *EMNLP*, pp. 7514–7528, 2021.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, pp. 6626–6637, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *CoRR*, abs/2207.12598, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33: 6840–6851, 2020.

Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models. *CoRR*, abs/2105.14080, 2021.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022.

Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2video-zero: Text-to-image diffusion models are zero-shot video generators. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 15908–15918. IEEE, 2023. doi: 10.1109/ICCV51070.2023.01462. URL https://doi.org/10.1109/ICCV51070.2023.01462.

Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. Bk-sdm: Architecturally compressed stable diffusion for efficient text-to-image generation. *ICML Workshop on Efficient Systems for Foundation Models (ES-FoMo)*, 2023.

Akio Kodaira, Chenfeng Xu, Toshiki Hazama, Takanori Yoshimoto, Kohei Ohno, Shogo Mitsuhori, Soichi Sugano, Hanying Cho, Zhijian Liu, and Kurt Keutzer. Streamdiffusion: A pipeline-level solution for real-time interactive generation. *arXiv*, 2023.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *ICLR*. OpenReview.net, 2021.

Yunsung Lee, Jin-Young Kim, Hyojun Go, Myeongho Jeong, Shinhyeok Oh, and Seungtaek Choi. Multi-architecture multi-expert diffusion models. *CoRR*, abs/2306.04990, 2023.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *CVPR*, pp. 991–999, 2015.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *ICML*, volume 202, pp. 19274–19286, 2023.

Senmao Li, Taihang Hu, Fahad Shahbaz Khan, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, and Jian Yang. Faster diffusion: Rethinking the role of unet encoder in diffusion models. *arXiv*, 2023a.

Xiuyu Li, Long Lian, Yijiang Liu, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-diffusion: Quantizing diffusion models. *ICCV*, 2023b.

Enshu Liu, Xuefei Ning, Zinan Lin, Huazhong Yang, and Yu Wang. OMS-DPM: optimizing the model schedule for diffusion probabilistic models. In *ICML*, volume 202, pp. 21915–21936. PMLR, 2023.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *ICLR*, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *NeurIPS*, 2022.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2023.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. *arXiv*, 2023.

Taehong Moon, Moonseok Choi, EungGu Yun, Jongmin Yoon, Gayoung Lee, and Juho Lee. Early exiting for accelerated inference in diffusion models. In *ICML 2023 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2023.

Zizheng Pan, Jianfei Cai, and Bohan Zhuang. Stitchable neural networks. In *CVPR*, 2023a.

Zizheng Pan, Jing Liu, Haoyu He, Jianfei Cai, and Bohan Zhuang. Stitched vits are flexible vision backbones. *arXiv*, 2023b.

William Peebles and Saining Xie. Scalable diffusion models with transformers. *CoRR*, abs/2212.09748, 2022.

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. *CoRR*, 2023.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*. OpenReview.net, 2023.

Geoffrey Roeder, Luke Metz, and Durk Kingma. On linear identifiability of learned representations. In *ICML*, volume 139, pp. 9030–9039, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pp. 10684–10695, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*. OpenReview.net, 2022.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv*, 2023.

Segmind. Segmind Stable Diffusion Model (SSD-1B). `https://huggingface.co/segmind/SSD-1B`, 2023.

Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. *CVPR*, Jun 2023.

Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. *CoRR*, abs/2305.16317, 2023.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*. OpenReview.net, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ICLR*, 2021b.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*, volume 202, 2023.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pp. 2818–2826. IEEE Computer Society, 2016.

Shengkun Tang, Yaqing Wang, Caiwen Ding, Yi Liang, Yao Li, and Dongkuan Xu. Deediff: Dynamic uncertainty-aware early exiting for accelerating diffusion model generation. *CoRR*, abs/2309.17074, 2023.

Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`, 2022.

Felix Wimbauer, Bichen Wu, Edgar Schönfeld, Xiaoliang Dai, Ji Hou, Zijian He, Artsiom Sanakoyeu, Peizhao Zhang, Sam S. Tsai, Jonas Kohler, Christian Rupprecht, Daniel Cremers, Peter Vajda, and Jialiang Wang. Cache me if you can: Accelerating diffusion models through block caching. *arXiv*, 2023.

Shuai Yang, Yukang Chen, Luozhou WANG, Shu Liu, and Ying-Cong Chen. Denoising diffusion step-aware models. In *ICLR*, 2024.

Xingyi Yang, Daquan Zhou, Songhua Liu, Jingwen Ye, and Xinchao Wang. Deep model reassembly. *NeurIPS*, 2022.

Xingyi Yang, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Diffusion probabilistic model made slim. In *CVPR*, pp. 22552–22562. IEEE, 2023.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, pp. 3836–3847, 2023.

Yang Zhao, Yanwu Xu, Zhisheng Xiao, and Tingbo Hou. Mobilediffusion: Subsecond text-to-image generation on mobile devices. *arXiv*, 2023.

Hongkai Zheng, Weili Nie, Arash Vahdat, Kamyar Azizzadenesheli, and Anima Anandkumar. Fast sampling of diffusion models via operator learning. In *ICML*, pp. 42390–42402. PMLR, 2023.

A Appendix

We organize our supplementary material as follows.

- In Section A.1, we introduce the preliminary of T-Stitch.
- In Section A.2, we provide guidelines for practical deployment of T-Stitch.
- In Section A.3, we provide frequency analysis during denoising sapmling process based on DiTs.
- In Section A.4, we report the details of our adopted pretrained DiTs and U-Nets.
- In Section A.5, we show the effect of using different classifier-free guidance scales based on DiTs and T-Stitch.

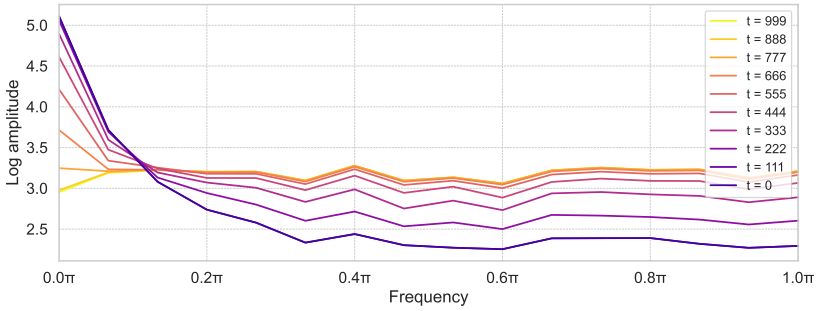


Figure 11: Frequency analysis in denoising process of DiT-XL, based on DDIM 10 steps and guidance scale of 4.0. We visualize the log amplitudes of Fourier-transformed latent noises at each step. Results are averaged over 32 images.

- In Section A.6, we compare FID evaluation under T-Stitch with 5,000 images and 50,000 images.
- In Section A.7, we compare T-Stitch with directly reducing sampling steps.
- In Section A.8, we show T-Stitch is compatible with pruned and knowledge distilled models.
- In Section A.9, we describe the implementation details of model stitching baseline under SN-Netv2 Pan et al. (2023b).
- In Section A.10, we show image examples when using T-Stitch with different sampling steps based on DiTs.
- In Section A.11, we demonstrate that T-Stitch is applicable to different pretrained model families, *e.g.*, stitching DiT with U-ViT Bao et al. (2023).
- In Section A.12, we show more image examples in stable diffusion experiments, including the original SDv1.4, stylized SDs, SDXL, ControlNet.
- In Section A.13, we report our finetune experiments by further finetuning the large DiTs at their allocated steps.
- In Section A.14, we report the precision and recall metrics on class conditional ImageNet-256 based on DiTs.
- In Section A.15, we show image examples of T-Stitch with DiTs and U-Nets.
- In Section A.16, we evaluate FID under T-Stitch by using pretrained DiT-S at different training iterations.
- In Section A.17, we demonstrate that T-Stitch can still obtain a smooth speed and quality trade-off under 2-4 steps with LCM Luo et al. (2023).
- In Section A.18, we show T-Stitch is also complementary to cache-based methods such as DeepCache Ma et al. (2023) to achieve further speedup.
- In Section A.19, we evaluate T-Stitch and show image examples by applying ToMe Bolya et al. (2023) simultaneously.
- In Section A.20, we show examples of using T-Stitch to accelerate zero-shot Text-to-Video generation Khachatryan et al. (2023).
- In Section A.21, we report the L2-distance of latent embeddings between DiTs at different denoising steps.
- In Section A.22, we discuss the relation with early-exit works.
- In Section A.23, we discuss the limitations of T-Stitch.

Table 6: Performance comparison of pretrained DiT model family on class-conditional ImageNet. FLOPs are measured by a single forward process given a latent noise in the shape of $4 \times 32 \times 32$.

	Parameters (M)	FLOPs (G)	Train Iters	Time Cost (s)	FID
DiT-S	33.0	5.5	5000K	1.6	33.46
DiT-B	130.5	21.8	1600K	4.0	12.30
DiT-XL	675.1	114.5	-	16.5	9.20

Table 7: Performance comparison of LDM and LDM-S on class-conditional ImageNet.

Model	Param (M)	Train Iter	Time Cost (s)	FID
LDM-S	25	400K	2.9	40.92
LDM	394	200K	7.1	20.11

A.1 PRELIMINARY

Diffusion models. We consider the class of score-based diffusion models in a continuous time (Song et al., 2021b) and following the presentation from Karras et al. (2022). Let $p_{data}(\mathbf{x}_0)$ denote the data distribution and $\sigma(t)\colon [0,1] \to \mathbb{R}_+$ is a user-specified noise level schedule, where $t \in \{0, ..., T\}$ and $\sigma(t-1) < \sigma(t)$. Let $p(\mathbf{x};\sigma)$ denote the distribution of noised samples by injecting σ^2-variance Gaussian noise. Starting with a high-variance Gaussian noise $\mathbf{x}_T$, diffusion models gradually denoise $\mathbf{x}_T$ into less noisy samples $\{\mathbf{x}_{T-1}, \mathbf{x}_{T-2}, ..., \mathbf{x}_0\}$, where $\mathbf{x}_t \sim p(\mathbf{x}_t; \sigma(t))$. Furthermore, this iterative process can be done by solving the probability flow ordinary differential equation (ODE) if knowing the score $\nabla_x \log p_t(x)$, namely the gradient of the log probability density with respect to data,

$$d\mathbf{x} = -\hat{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x};\sigma(t))\, dt, \tag{1}$$

where $\hat{\sigma}(t)$ denote the time derivative of $\sigma(t)$. Essentially, diffusion models aim to learn a model for the score function, which can be reparameterized as

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \approx (D_\theta(\mathbf{x};\sigma) - \mathbf{x})/\sigma^2, \tag{2}$$

where $D_\theta(\mathbf{x};\sigma)$ is the learnable denoiser. Given a noisy data point $\mathbf{x}_0 + \mathbf{n}$ and a conditioning signal $\boldsymbol{c}$, where $\boldsymbol{n} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \boldsymbol{I}\right)$, the denoiser aim to predict the clean data $\mathbf{x}_0$. In practice, the mode is trained by minimizing the loss of denoising score matching,

$$\mathbb{E}_{(\mathbf{x}_0,\mathbf{c})\sim p_{\text{data}},(\sigma,\mathbf{n})\sim p(\sigma,\mathbf{n})}\left[\lambda_\sigma \|D_{\boldsymbol{\theta}}(\mathbf{x}_0 + \mathbf{n};\sigma,\mathbf{c}) - \mathbf{x}_0\|_2^2\right], \tag{3}$$

where $\lambda_\sigma\colon \mathbb{R}_+ \to \mathbb{R}_+$ is a weighting function (Ho et al., 2020), $p(\sigma,\mathbf{n}) = p(\sigma)\mathcal{N}\left(\mathbf{n};\mathbf{0},\sigma^2\right)$, and $p(\sigma)$ is a distribution over noise levels σ.

This work focuses on *the denoisers* D in diffusion models. In common practice, they are typically large parameterized neural networks with different architectures that consume high FLOPs at each timestep. In the following, we use "denoiser" or "model" interchangeably to refer to this network. We begin with the pretrained DiT model family to explore the advantage of trajectory stitching on efficiency gain. Then we show our method is a general technique for other architectures, such as U-Net (Rombach et al., 2022) and U-ViT (Bao et al., 2023).

Classifier-free guidance. Unlike classifier-based denoisers (Dhariwal & Nichol, 2021) that require an additional network to provide conditioning guidance, classifier-free guidance (Ho & Salimans, 2022) is a technique that jointly trains a conditional model and an unconditional model in one network by replacing the conditioning signal with a null embedding. During sample generation, it adopts a guidance scale $s \geq 0$ to guide the sample to be more aligned with the conditioning signal by jointly considering the predictions from both conditional and unconditional models,

$$D^s(\mathbf{x};\sigma,\mathbf{c}) = (1+s)D(\mathbf{x};\sigma,\mathbf{c}) - sD(\mathbf{x};\sigma). \tag{4}$$

Recent works have demonstrated that classifier-free guidance provides a clear improvement in generation quality. In this work, we consider the diffusion models that are trained with classifier-free guidance due to their popularity.

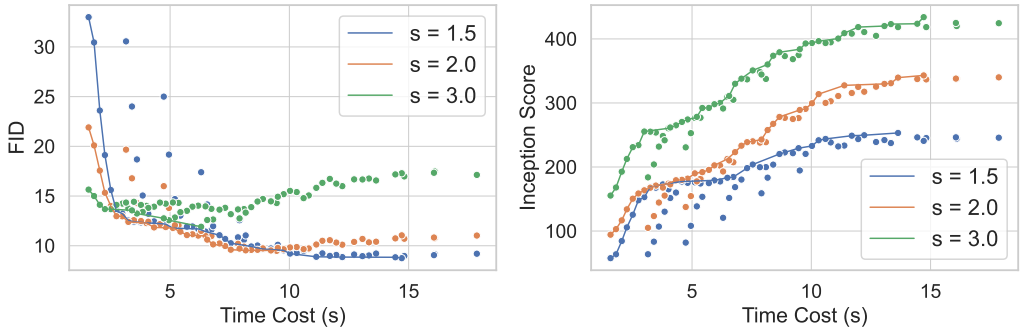


Figure 12: Trajectory stitching based on three models: DiT-S, DiT-B, and DiT-XL. We adopt DDIM 100 timesteps with a classifier-free guidance scale of 1.5, 2.0 and 3.0.

A.2 Practical Deployment of T-Stitch

In this section, we provide guidelines for the practical deployment of T-Stitch by formulating our model allocation strategy into a compute budget allocation problem.

Given a set of denoisers $\{D_1, D_2, ..., D_K\}$ and their corresponding computational costs $\{C_1, C_2, ..., C_K\}$ for sampling in a T-steps trajectory, where $C_{k-1} < C_k$, we aim to find an optimal configuration set $\{r_1, r_2, ..., r_K\}$ that allocates models into corresponding denoising intervals to maximize the generation quality, which can be formulated as

$$\max_{r_1, r_2, ..., r_K} \quad M\left(F(D_1, r_1) \circ F(D_2, r_2) \cdots \circ F(D_K, r_K)\right) \tag{5}$$

$$\text{subject to} \quad \sum_{k=1}^{K} r_k C_k \leq C_R, \sum_{k=1}^{K} r_k = 1, \tag{6}$$

where $F(D_k, r_k)$ refers to the denoising process by applying denoiser D_k at the k-th interval indicated by r_k, $\circ$ denotes to a composition, M represents a metric function for evaluating generation performance, and C_R is the compute budget constraint. Since $\{C_1, C_2, ..., C_K\}$ is known, we can efficiently enumerate all possible fraction combinations and obtain a lookup table, where each fraction configuration set corresponds to a compute budget (*i.e.*, time cost). In practice, we can sample a few configuration sets from this table that satisfy a budget and then apply to generation tasks.

A.3 Frequency Analysis in Denoising Process

We provide evidence that the denoising process focuses on low frequencies at the initial stage and high frequencies in the later steps. Based on DiT-XL, we visualize the log amplitudes of Fourier-transformed latent noises at each sampling step. As shown in Figure 11, the low-frequency amplitudes increase rapidly at the early timesteps (*i.e.*, from 999 to 555), indicating that low frequencies are intensively generated. At the later steps, especially for $t = 111$ and $t = 0$, we observe the log amplitude of high frequencies increases significantly, which implies that the later steps focus on detail refinement.

A.4 Pretrained DiTs and U-Nets

In Table 6 and Table 7, we provide detailed comparisons of the pretrained DiT model family, as well as our reproduced small version of U-Net. Overall, as mentioned earlier in Section 3, we make sure the models at each model family have a clear gap in model size between each other such that we can achieve a clear speedup.

A.5 Effect of Different Classifier-free Guidance on Three-model T-Stitch

In Figure 12, we provide the results by applying T-Sittch with DiTs using different guidance scales under three-model settings. In general, T-Stitch performs consistently with different guidance scales,

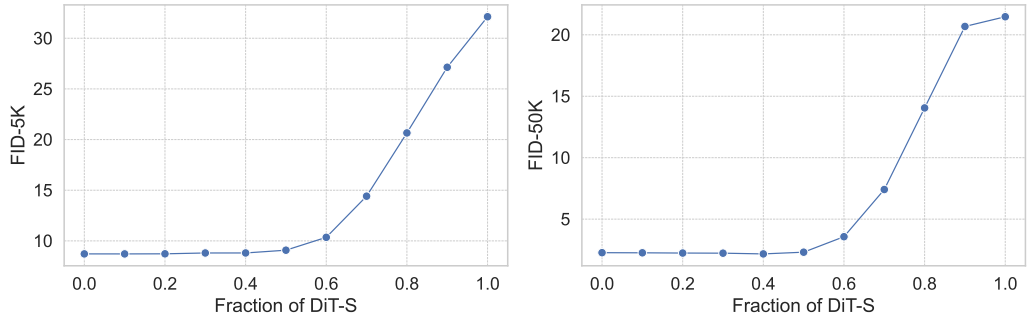


Figure 13: Trajectory stitching based on three models: DiT-S, DiT-B, and DiT-XL. We adopt DDPM 250 timesteps with a classifier-free guidance scale of 1.5.

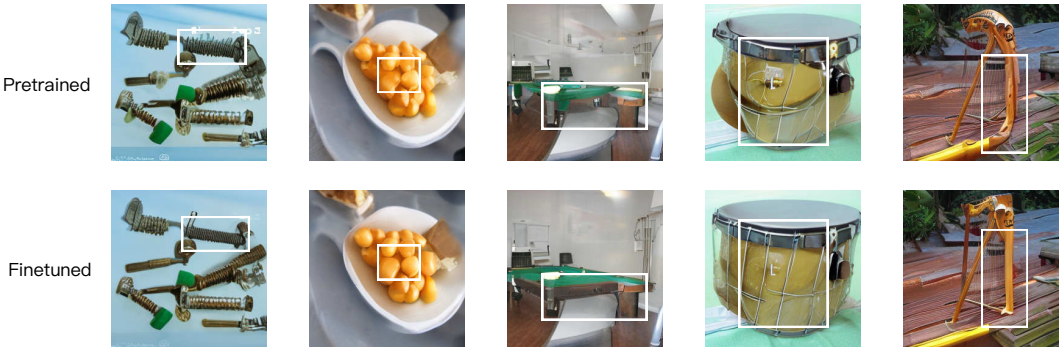


Figure 14: Image quality comparison by stitching pretrained and finetuned DiT-B and DiT-XL at the later steps, based on T-Stitch schedule of DiT-S/B/XL of 50% : 30% : 20%.

where it interpolates a smooth Pareto frontier between the DiT-S and DiT-XL. As common practice in DPMs adopt different guidance scales to control image generation, this significantly underscores the broad applicability of T-Stitch.

A.6 FID-50K vs. FID-5K

For efficiency concerns, we report FID based on 5,000 images by default. Based on DiT, we apply T-Stitch with DDPM 250 steps with a guidance scale of 1.5 and sample 50,000 images for evaluating FID. As shown in Figure 13, the observation between FID-50K and FID-5K are similar, which indicates that sampling more images like 50,000 does not affect the effectiveness.

A.7 Compared to Directly Reducing Sampling Steps

Reducing sampling steps has been a common practice for obtaining different speed and quality trade-offs during deployment. Although we have demonstrated that T-Stitch can achieve consistent efficiency gain under different sampling steps, we show in Figure 15 that compared to directly reducing the number of sampling steps, the trade-offs from T-Stitch are very competitive, especially for the 50-100 steps region where the FIDs under T-Stitch are even better. Thus, T-Stitch is able to serve as a *complementary* or an alternative method for practical DPM sampling speed acceleration.

A.8 Compared to Model Compression

In practice, T-Stitch is orthogonal to individual model optimization/compression. For example, with a BK-SDM Tiny and SDv1.4, we can still apply compression into SDv1.4 in order to reduce the computational cost at the later steps from the large SD. In Figure 16, we show that by adopting a

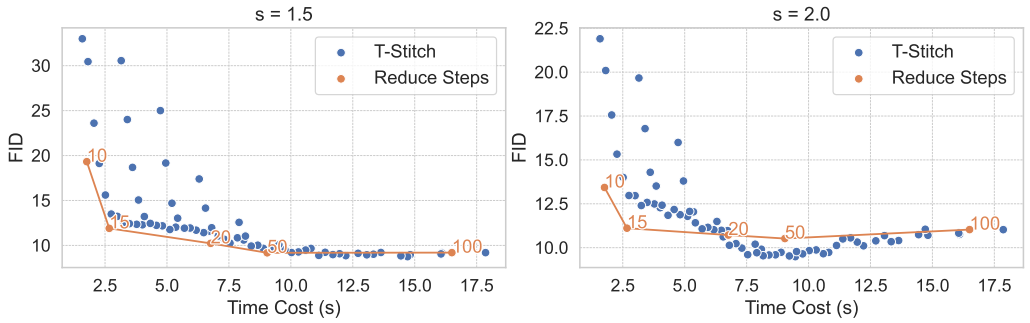


Figure 15: Based on DDIM, we report the FID and speedup comparisons on DiT-XL by using T-Stitch and directly reducing the sampling step from 100 to 10. "s" denotes the classifier-free guidance scale. Trajectory stitching adopts the three-model combination (DiT-S/B/XL) under 100 steps.

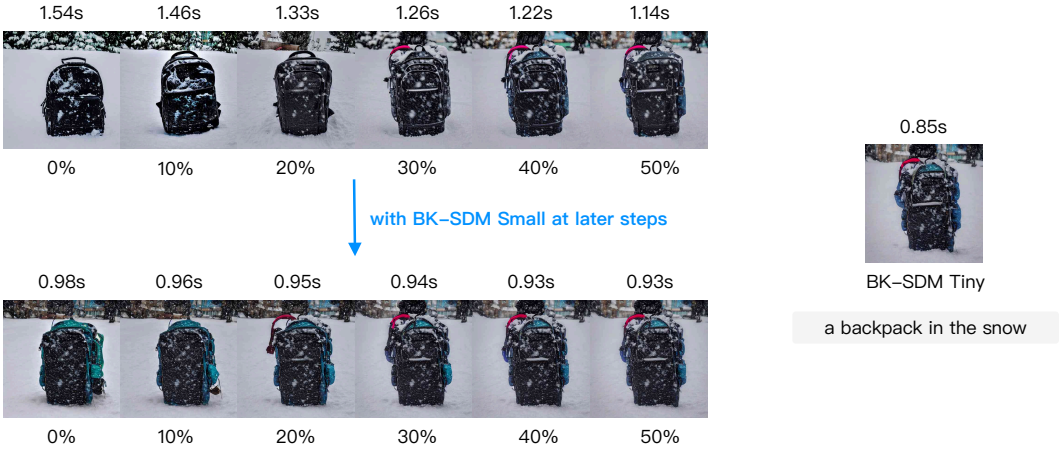


Figure 16: Comparison of T-Stitch by adopting SDv1.4 and its compressed version (*i.e.*, BK-SDM Small) at the later steps.

compressed SD v1.4, *i.e.*, BK-SDM Small, we can further reduce the time cost with a trade-off for image quality.

A.9 Implementation Details of Model Stitching Baseline

We adopt a LoRA rank of 64 when stitching DiT-S/XL, which leads to 134 stitching configurations. The stitched model is finetuned on 8 A100 GPUs for 1,700K training iterations. We pre-extract the ImageNet features with a Stable Diffusion AutoEncoder (Rombach et al., 2022) and do not apply any data augmentation. Following the baseline DiT, we adopt the AdamW optimizer with a constant learning rate of 1×10^{-4}. The total batch size is set as 256. All other hyperparameters adopt the default setting as DiT.

A.10 Image Examples under the Different Number of Sampling Steps

Figure 17 shows image examples generated using different numbers of sampling steps under T-Stitch and DiT-S/XL. As the figure shows, adopting a small model at the early 40% steps has a negligible effect on the final generated images. When progressively increasing the fraction of DiT-S, there is a visible trade-off between speed and quality, with the final image becoming more similar to those generated from DiT-S.

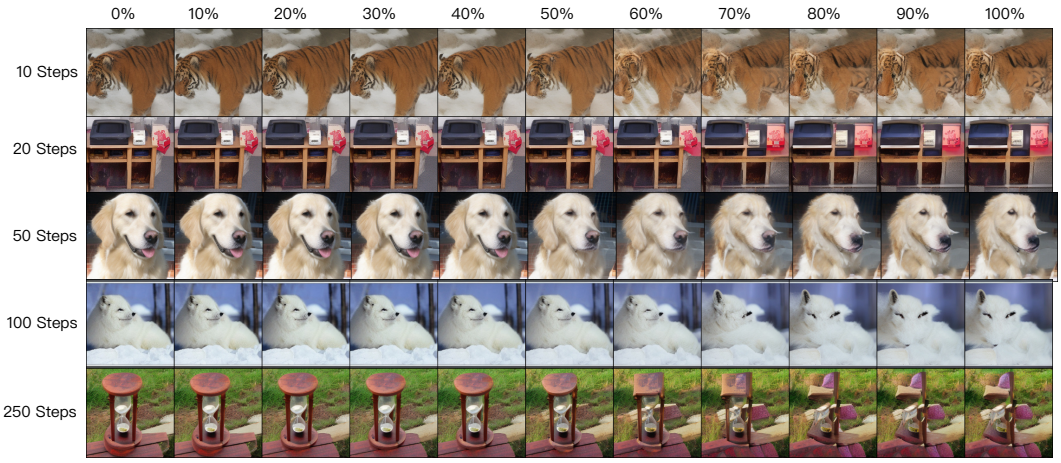


Figure 17: Based on DDIM and a classifier-free guidance scale of 1.5, we stitch the trajectories from DiT-S and DiT-XL and progressively increase the fraction (%) of DiT-S timesteps at the beginning.

Table 8: T-Stitch with DiT-S and U-ViT H, under DPM-Solver++, 50 steps, guidance scale of 1.5.

Fraction of DiT-S	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
FID	15.04	13.68	12.44	12.76	14.19	17.6	29.4	53.75	74.14	84.33	121.95
Time Cost (s)	15.90	13.21	11.91	10.61	9.42	7.92	6.57	5.23	3.84	2.50	1.40

A.11 T-Stitch with Different Pretrained Model families

As different pretrained models trained on the same dataset to learn similar encodings, T-Stitch is able to directly integrate different pretrained model families. For example, based on U-ViT H (Bao et al., 2023), we apply DiT-S at the early sampling steps just as we have done for DiTs and U-Nets. Remarkably, as shown in Table 8, it performs very well, which demonstrates the advantage of T-Stitch as it can be applied for more different models in the public model zoo.

A.12 More Examples in Stable Diffusion

We show more examples by applying T-Stitch to SD v1.4, InkPunk Diffusion and Ghibli Diffusion with a small SD model, BK-SDM Tiny (Kim et al., 2023). For all examples, we adopt the default scheduler and hyperparameters of StableDiffusionPipeline in Diffusers: PNDM scheduler, 50 steps, guidance scale 7.5. In Figure 25, we observe that adopting a small SD in the sampling trajectory of SD v1.4 achieves minor effect on image quality at the small fractions and obtain flexible trade-offs in speed and quality by using different fractions.

Stylized SDs. For stylized SDs, such as InkPunk-Diffusion and Ghibli-Diffusion[2], we show in Figures 26 and 27 that T-Stitch helps to complement the prompt alignment by effectively utilizing the knowledge of the pretrained small SD. Benefiting from the interpolation on speeds, styles and image contents, T-Stitch naturally increases the diversity of the generated images given a prompt by using different fractions of small SD.

Generality of T-Stitch . In Figure 28, we show T-Stitch performs favorably with more complex prompts. Besides, by adopting a smaller and distilled SSD-1B, we can easily accelerate SDXL while being compatible with complex prompts and ControlNet (Zhang et al., 2023) for practical art generation, as shown in Figure 29 and Figure 30. Furthermore, we demonstrate that T-Stitch is robust in practical usage. As shown in Figure 31, 8 consecutive runs can generate stable images with great quality.

[2]https://huggingface.co/nitrosocke/Ghibli-Diffusion

Table 10: Precision and Recall evaluation based on DiT-S/XL, with DDIM 100 steps and guidance scale of 1.5.

Fraction of DiT-S	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
FID	9.20	9.17	8.99	9.03	8.95	10.06	12.46	18.04	25.44	30.11	33.46
Precision	0.81	0.81	0.81	0.81	0.80	0.76	0.72	0.67	0.62	0.59	0.58
Recall	0.74	0.74	0.74	0.74	0.75	0.75	0.74	0.73	0.69	0.65	0.63

A.13 Finetuning on Specific Trajectory Schedule

Table 9: Performance comparison of stitching pretrained and finetuned DiTs at the later steps. We set the denoising interval of DiT-S/B/XL with 50% : 30% : 20%

	FID	Inception Score
Pretrained	16.49	123.11
Finetuned at all timesteps	16.04	125.81
Finetuned at stitched interval	13.35	155.35

When progressively using a small model in the trajectory, we observe a non-negligible performance drop. However, we show that we can simply finetune the model at the allocated denoising intervals to improve the generation quality. For example, based on DDIM and 100 steps, allocating DiT-S at the early 50%, DiT-B at the subsequent 30%, and DiT-XL at the last 20% obtains an FID of 16.49. In this experiment, we separately finetune DiT-B and DiT-XL **at their allocated denoising intervals**, with additional 250K iterations on ImageNet-1K under the default hyperparameters in DiT (Peebles & Xie, 2022). In Table 9, we observe a clear improvement over FID, Precision and Recall by finetuning at stitched interval. This strategy also achieves better performance than finetuning for all timesteps. Furthermore, we provide a comparison of the generated images in Figure 14, where we observe that finetuning clearly improves local details.

A.14 Precision and Recall Measurement of T-Stitch

Following common practice (Dhariwal & Nichol, 2021), we adopt Precision to measure fidelity and Recall to measure diversity or distribution coverage. In Table 10, we show that T-Stitch introduces a minor effect on Precision and Recall at the early 40-50% steps, while at the later steps we observe clear trade-offs, which is consistent with FID evaluations.

A.15 Image Examples of T-Stitch on DiTs and U-Nets

In Figures 23 and 24, we provide image examples that generated by applying T-Stitch with DiT-S/XL, LDM-S/LDM, respectively. Overall, we observe that adopting a small DPM at the beginning still produces meaningful and high-quality images, while at the later steps it achieves flexible speed and quality trade-offs. Note that different from DiTs that learn a null class embedding during classifier-free guidance, LDM inherently omits this embedding in their official implementation [3]. During sampling, LDM and LDM-S have different unconditional signals, which eventually results in various image contents under different fractions.

A.16 Effect of DiT-S under Different Training Iterations

In our experiments, we adopt a DiT-S that trained with 5,000K iterations as it can be sufficiently optimized. In Figure 18, we indicate that even under a short training schedule of 400K iterations, adopting DiT-S at the initial stages of the sampling trajectory also has a minor effect on the overall FID. The main difference is at the later part of the sampling trajectory. Therefore, it implies the early denoising sampling steps can be easier to learn and be handled by a compute-efficient small model.

A.17 Compatibility with LCM

T-Stitch can further speed up an already accelerated DPM via established training-based methods, such as step distillations (Luo et al., 2023; Song et al., 2023). For example, as shown in Figure 32

[3] https://github.com/CompVis/latent-diffusion

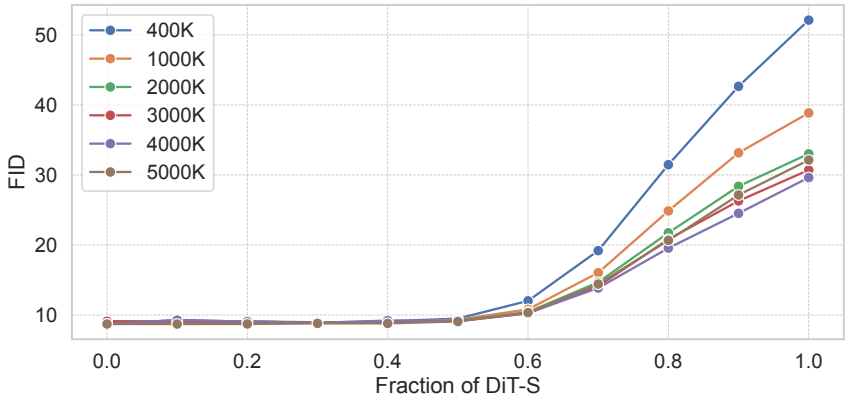


Figure 18: Effect of different pretrained DiT-S in T-Stitch for accelerating DiT-XL, based on DDPM, 250 steps and guidance scale of 1.5. For example, "400K" indicates the pretrained weights of DiT-S at 400K iterations.

and Figure 33, given a distilled SDXL from LCM (Luo et al., 2023), T-Stitch can achieve further speedup under 2 to 4 steps with high image quality by adopting a relatively smaller SD. In Table 11, Table 12, we report comprehensive FID, inception score and CLIP score evaluations by stitching LCM distilled SDXL and SSD-1B, where we show that T-Stitch smoothly interpolates the quality between SDXL and SSD-1B. Finally, we assume a better and faster small model in T-Stitch will help to obtain more gains in future works.

Table 11: T-Stitch based on LCM (Luo et al., 2023) distilled models: LCM-SDXL and LCM-SSD-1B, under 2 sampling steps.

Faction of SSD-1B	0	0.5	1
FID	21.98	23.96	24.36
IS	28.48	27.60	28.22
CLIP Score	0.2929	0.2895	0.2844
Time Cost (ms)	768	685	634

A.18 Compatibility with DeepCache

In this section, we demonstrate that recent cache-based methods Ma et al. (2023); Wimbauer et al. (2023) such as DeepCache Ma et al. (2023) can be effectively combined with T-Stitch to obtain more benefit. Essentially, as T-Stitch directly drops off the pretrained SDs, we can adopt DeepCache to simultaneously accelerate both small and large diffusion models during sampling to achieve further speedup. The image quality and speed benchmarking as shown in Figure 21 have demonstrated that T-Stitch works very well along with DeepCache, while potentially further improving the prompt alignment for stylized SDs. We also comprehensively evaluate the FID, Inception score, CLIP score and time cost in Figure 19, where we observe combining T-Stitch with DeepCache brings

Table 12: T-Stitch based on LCM (Luo et al., 2023) distilled models: LCM-SDXL and LCM-SSD-1B, under 4 sampling steps. Time cost is measured by generating one image on RTX 3090 in seconds.

Faction of SSD-1B	0%	25%	50%	75%	100%
FID	17.05	18.32	21.28	23.67	25.50
IS	35.35	34.46	31.83	30.47	29.31
CLIP Score	0.3062	0.3059	0.2984	0.2912	0.2897
Time Cost (ms)	1,029	968	921	870	823

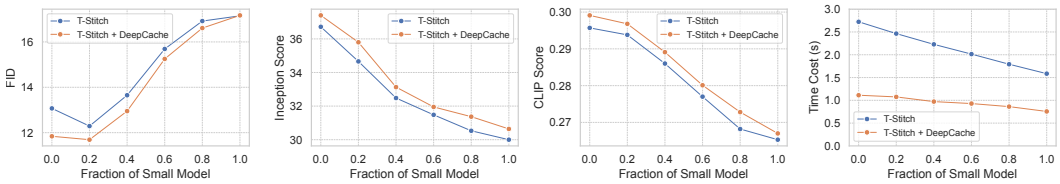


Figure 19: Effect of combining T-Stitch and DeepCache Ma et al. (2023). We report FID, Inception Score and CLIP score (Hessel et al., 2021) on MS-COCO 256×256 benchmark under 50 steps. The time cost is measured by generating one image on one RTX 3090. We adopt BK-SDM Tiny and SDv1.4 as the small and large model, respectively. For DeepCache, we adopt an uniform cache interval of 3.

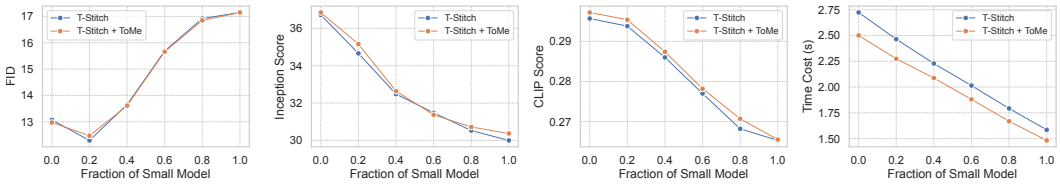


Figure 20: Effect of combining T-Stitch and ToMe Bolya et al. (2023). We report FID, Inception Score and CLIP score (Hessel et al., 2021) on MS-COCO 256×256 benchmark under 50 steps. The time cost is measured by generating one image on one RTX 3090. We adopt BK-SDM Tiny and SDv1.4 as the small and large model, respectively. For ToMe, we adopt a token merging ratio of 0.5.

improvement over all metrics. Note that under DeepCache, BK-SDM Tiny is 1.5× faster than SDv1.4, thus the speedup gain from T-Stitch is slightly smaller than applying T-Stitch only where the BK-SDM Tiny is 1.7× faster than SDv1.4. In addition, we observe DeepCache cannot work well with step-distilled models and ControlNet, while T-Stitch is generally applicable to many scenarios, as shown in Section A.12.

A.19 Compatibility with Token Merging

Our technique also complements Token Merging Bolya et al. (2023). For example, during the denoising sampling, we can still apply ToMe into both small and large U-Nets. In practice, it brings additional gain in both sampling speed and CLIP score, and slightly improves Inception score, as shown in Figure 20. We also provide image examples in Figure 22.

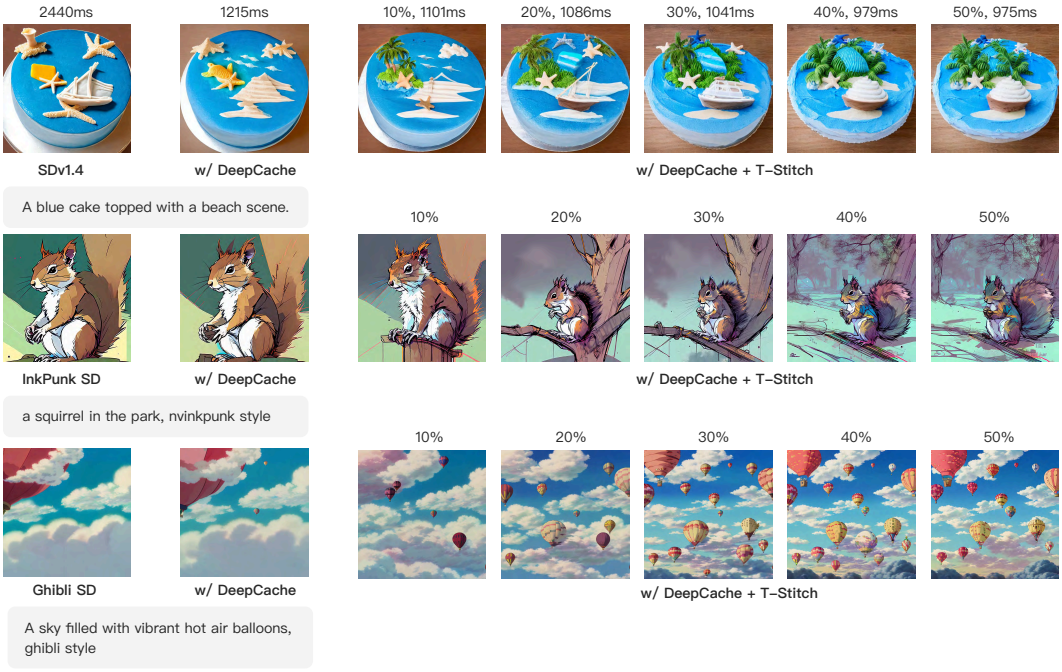


Figure 21: Image examples of combining T-Stitch with DeepCache Ma et al. (2023). We adopt BK-SDM Tiny as the small model in T-Stitch and report the percentage on the top of images. All images are generated by the default settings in diffusers von Platen et al. (2022): 50 steps with a guidance scale of 7.5.

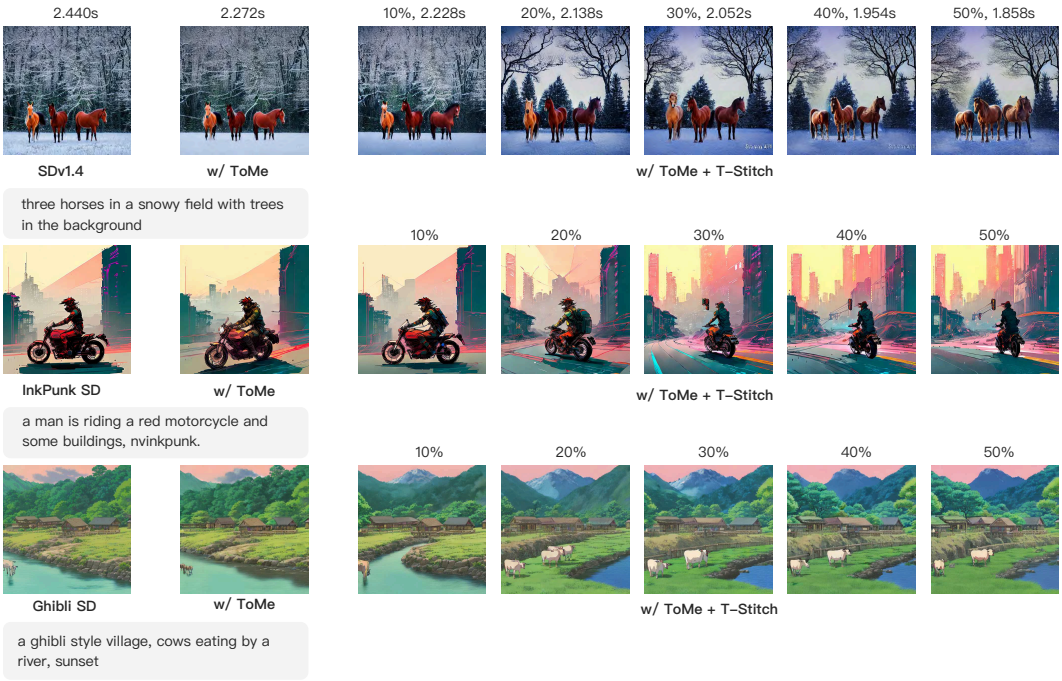


Figure 22: Image examples of combining T-Stitch and ToMe Bolya et al. (2023). We adopt BK-SDM Tiny as the small model in T-Stitch and report the percentage on the top of images. All images are generated by the default settings in diffusers von Platen et al. (2022): 50 steps with a guidance scale of 7.5. We adopt a token merging ratio of 0.5 in ToMe.

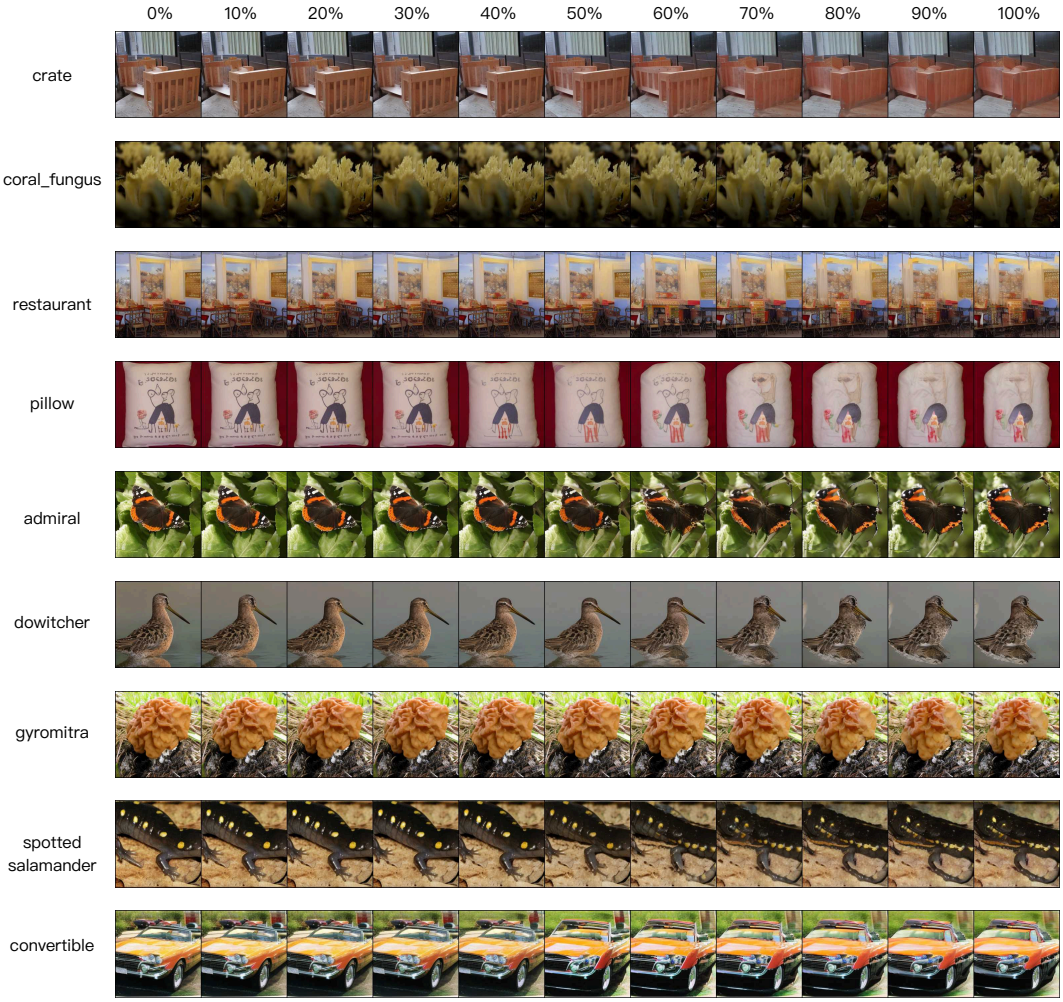


Figure 23: Image examples of T-Stitch on DiT-S and DiT-XL. We adopt DDIM and 100 steps, with a guidance scale of 4.0. From left to right, we gradually increase the fraction of LDM-S steps at the beginning, then let the original LDM to process later denoising steps.

A.20 Experiments on Zero-shot Text-to-Video Generation

In this section, we show T-Stitch works very well with text-to-video generation. Based on Text2Video-Zero Khachatryan et al. (2023), we found T-Stitch can also speedup video generation by adopting a small model like BK-SDM Tiny. We also observed a slight improvement on prompt alignment. For example, the "outer space" in Figure 34, where we progressively allocate BK-SDM Tiny at the 10-50% early sampling steps.

A.21 L2-Distance Comparison of Latent Emebddings for DiTs

As a complementary analysis to Figure 3, we present the L2-distance comparison of latent embeddings across different steps for DiTs in Figure 35. Overall, we observe that the latent embeddings from small and large DiTs are notably more similar during the early steps compared to the later steps, aligning with the patterns observed in Figure 3.

A.22 Comparison with Early-exit Works

In general, we aim to explore the compute budget allocation for diffusion model sampling, which is orthogonal with individual model acceleration techniques such as early-exiting or model compres-

Figure 24: Image examples of T-Stitch on U-Net-based LDM and LDM-S. We adopt DDIM and 100 steps, with a guidance scale of 3.0. From left to right, we gradually increase the fraction of LDM-S steps at the beginning, then let the original LDM to process later denoising steps.

sion that specifically focus on one model, as discussed in Section A.8. We provide a short discussion with early-exit works as below.

Deediff Tang et al. (2023) introduces Timestep-Aware Uncertainty Estimation Module (UEM), which includes new trainable parameters. Compared to it, T-Stitch does not require any new parameters to train since we directly drop the small model at the early denoising steps. Adaptive Score Estimation (ASE) Moon et al. (2023) heuristically designs block-exiting strategies based on different architectures, then finetuning the target diffusion model with substantial training cost. Based on DiT-XL, we found our speed-quality trade-offs are clearly better, as shown below.

Table 13: Comparison of DiT models in FID-5K scores and acceleration gains, as reported in Moon et al. (2023).

Name	**FID-5K**	**Acceleration**
DiT-XL (Moon et al. (2023) implementation)	9.10	-
D1-DiT	8.89	14.38%
D3-DiT	8.99	20.99%
D4-DiT	9.19	28.70%
D6-DiT	11.41	36.80%

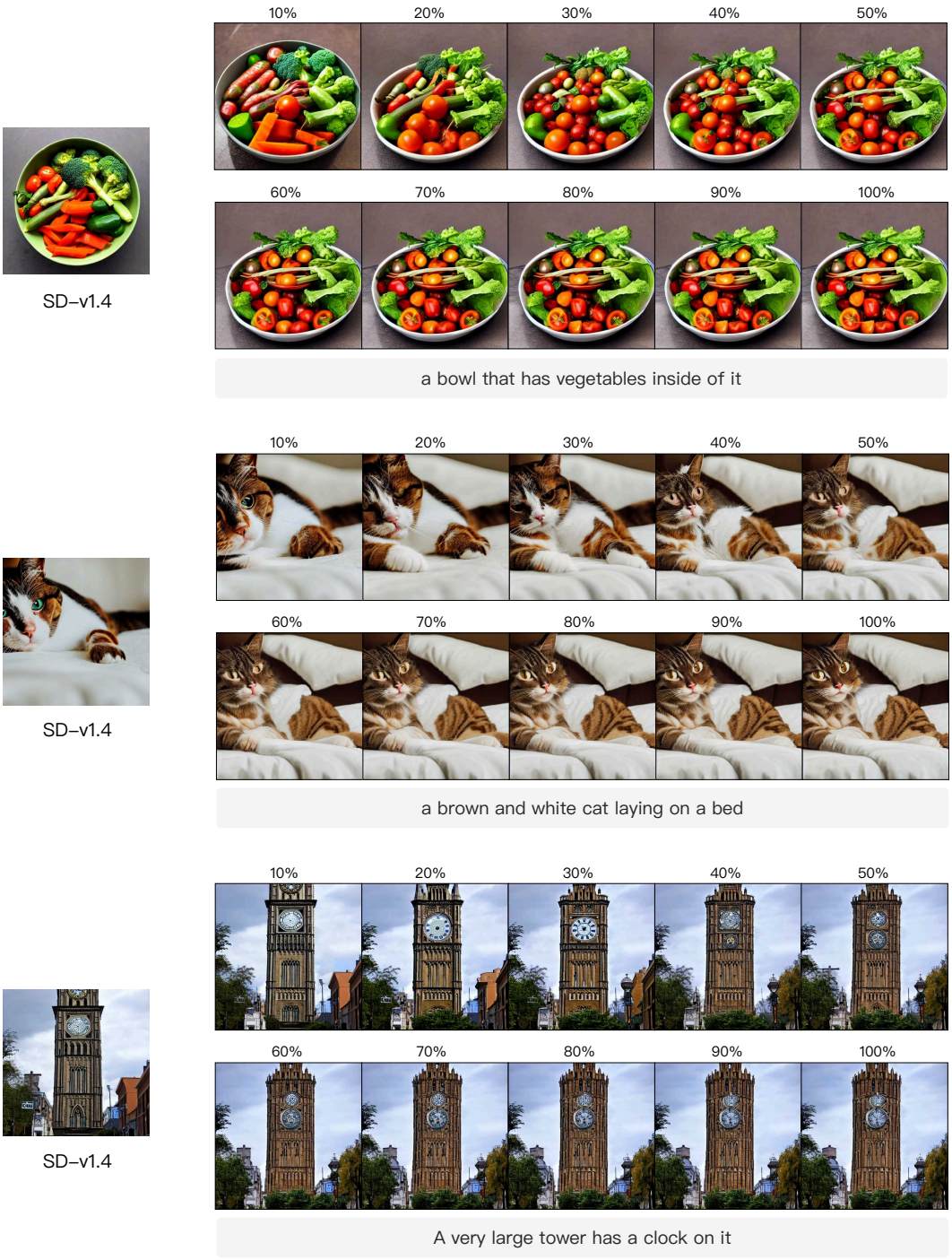


Figure 25: T-Stitch based on Stable Diffusion v1.4 and BK-SDM Tiny. We annotate the faction of BK-SDM on top of images.

In Table 13, we briefly summarize the main results in Moon et al. (2023). FID-5K is evaluated based on ImageNet-256 and DDIM sampler. "Acceleration" refers to the acceleration in sampling speed. "n" in "Dn-DiT" represents the acceleration scale. Details for different settings can be found in Table 2 of ASE Moon et al. (2023).

The results in Table 14 are from our Figure 1, which is based on the same experimental setting: ImageNet-256, DDIM sampler, and FID-5K. Note that due to different implementations, our DiT-

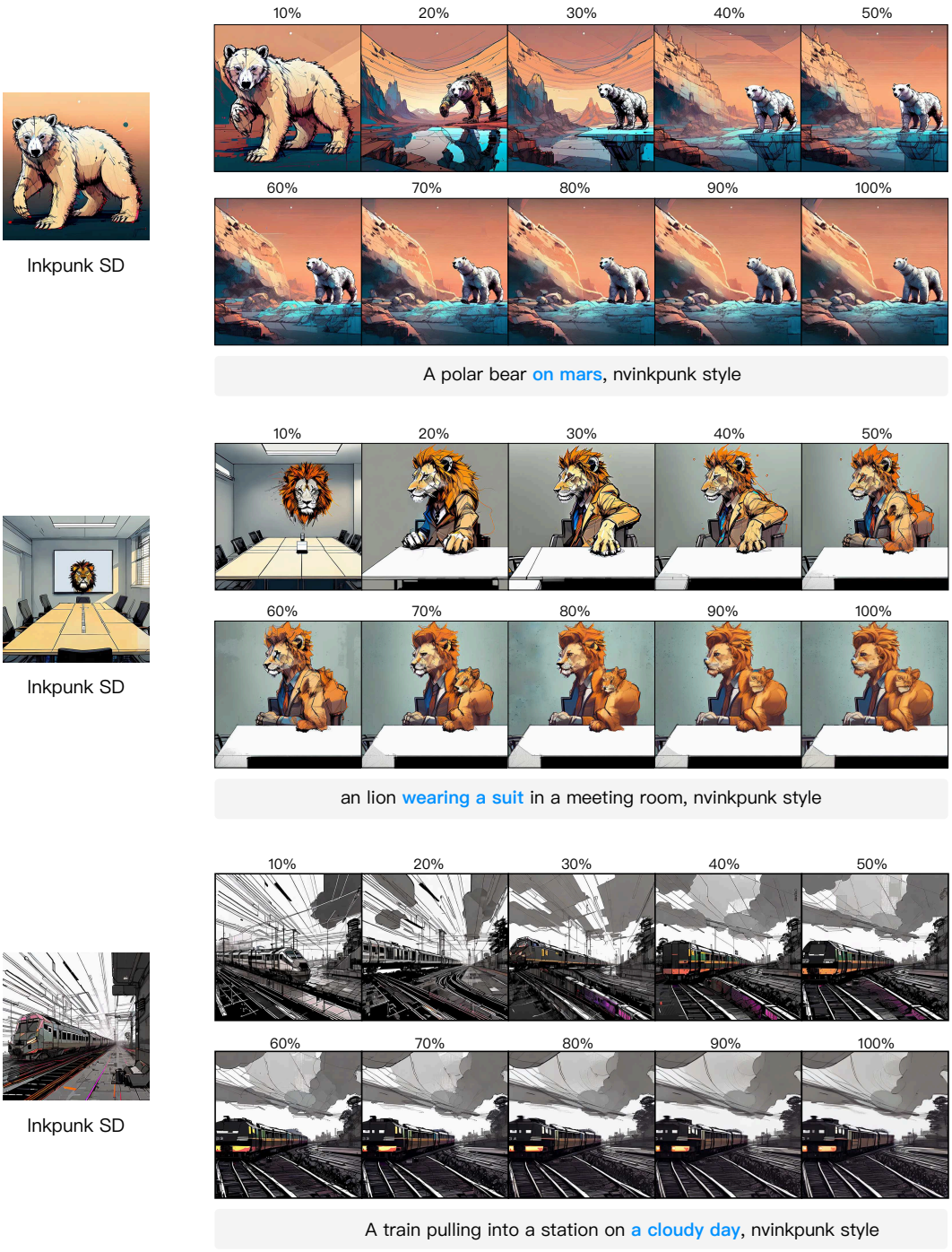


Figure 26: T-Stitch based on Inkpunk-Diffusion SD an BK-SDM Tiny. We annotate the faction of BK-SDM on top of images.

XL baseline performance can be slightly different. Furthermore, compared to these training-based approaches, T-Stitch is training-free and can be directly applicable to many various scenarios.

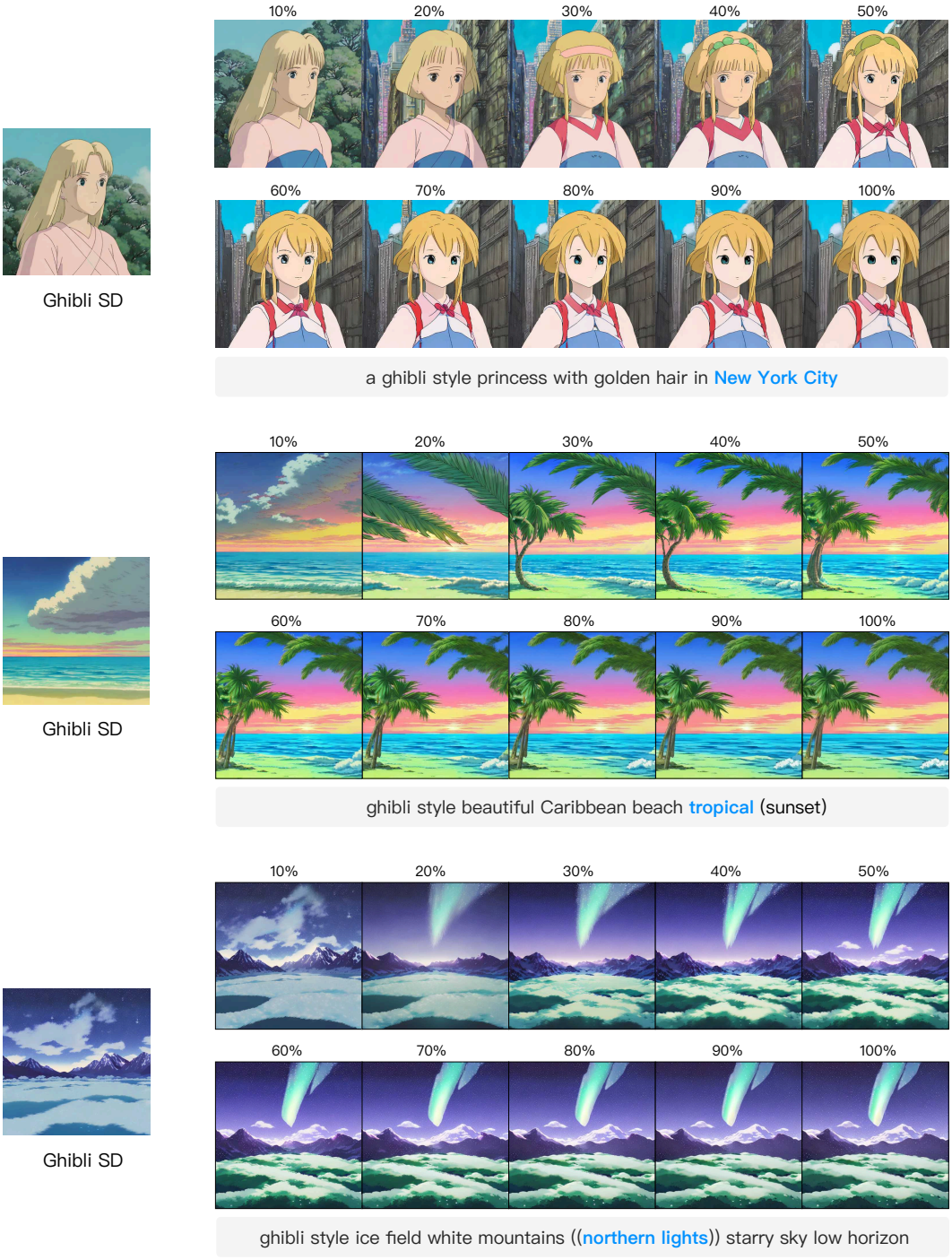


Figure 27: T-Stitch based on Ghibli-Diffusion SD and BK-SDM Tiny. We annotate the faction of BK-SDM on top of images.

A.23 LIMITATIONS

T-Stitch requires a smaller model that has been trained on the same data distribution as the large model. Thus, a sufficiently optimized small model that shares the similar latent space as the target large model is required. Besides, adopting an additional small model for denoising sampling will *slightly* increase memory usage (Table 4). Lastly, since T-Stitch provides a free lunch from a small

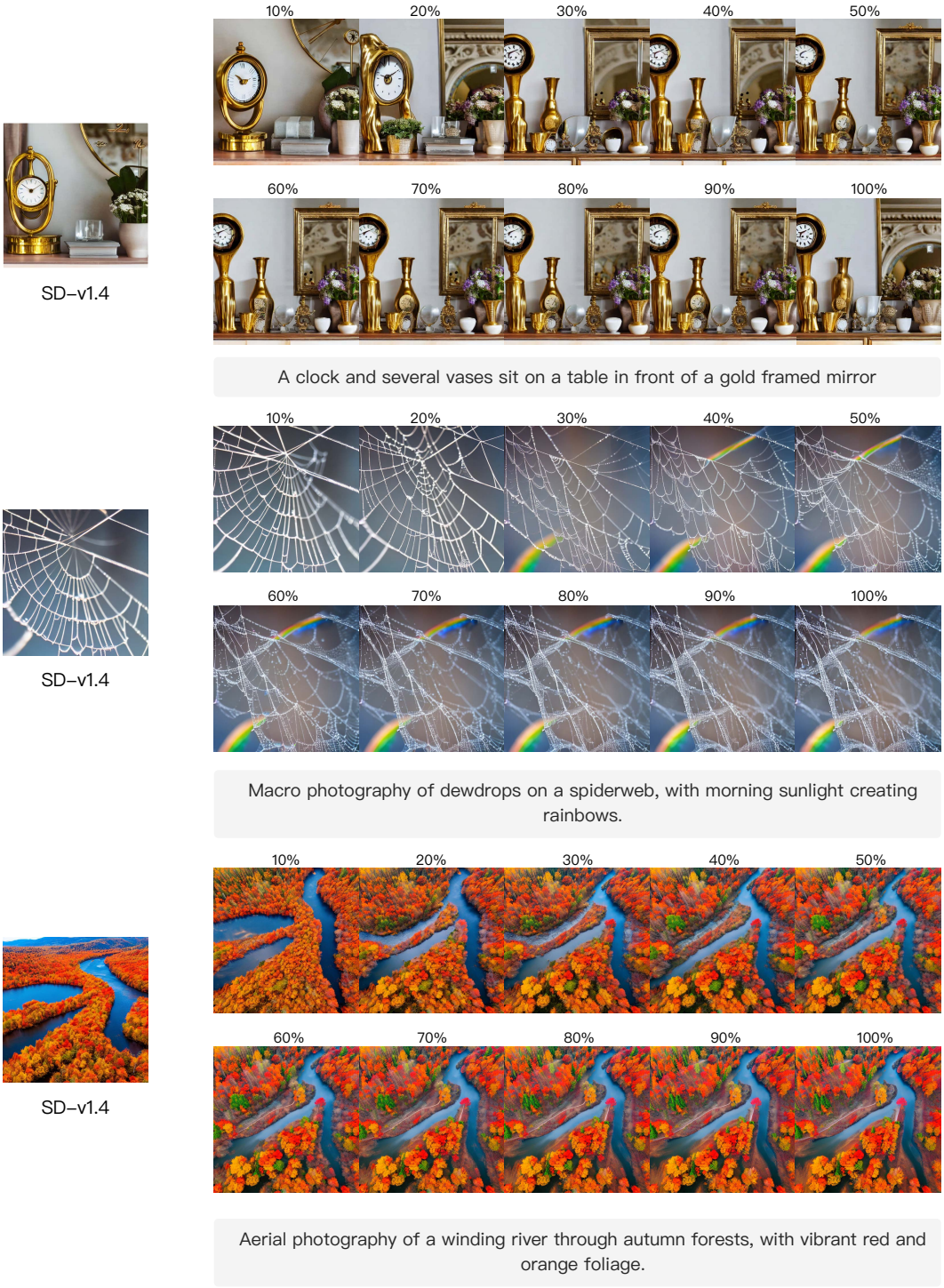


Figure 28: T-Stitch with more complex prompts based on Stable Diffusion v1.4 and BK-SDM Tiny. We annotate the faction of BK-SDM on top of images.

model for sampling acceleration, the speedup gain is bounded by the efficiency of the small model. In practice, we suggest using T-Stitch when a small model is available and much faster than the large model. Furthermore, when applying T-Stitch with very different architectures, the main challenge is that the two model may have different spatial dimension for their latents, thus making it difficult to directly switching models during sampling. We leave better stitching strategies for future works.

Prompt: highly detailed albert einstein at an epic laboratory office, shelves with detailed items in background, ((long shot)), highly detailed realistic painting by grandmaster, unreal engine, octane render, 4k, trending on artstation

Prompt: concept art of dragon flying over town, clouds. digital artwork, illustrative, painterly, matte painting, highly detailed, cinematic composition

Negative Prompt: photo, photorealistic, realism, ugly

Prompt: anime artwork an empty classroom. anime style, key visual, vibrant, studio anime, highly detailed

Negative Prompt: photo, deformed, black and white, realism, disfigured, low contrast

Prompt: claymation style captain jack sparrow on tropical island. sculpture, clay art, centered composition, play–doh

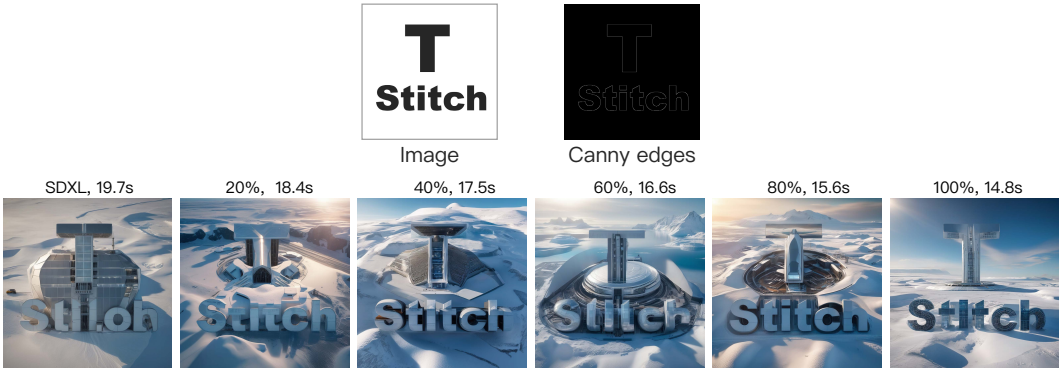
Negative Prompt: sloppy, messy, grainy, highly detailed, ultra textured, photo, mutated

Prompt: 16–bit pixel art, a cozy cafe side view, a beautiful day

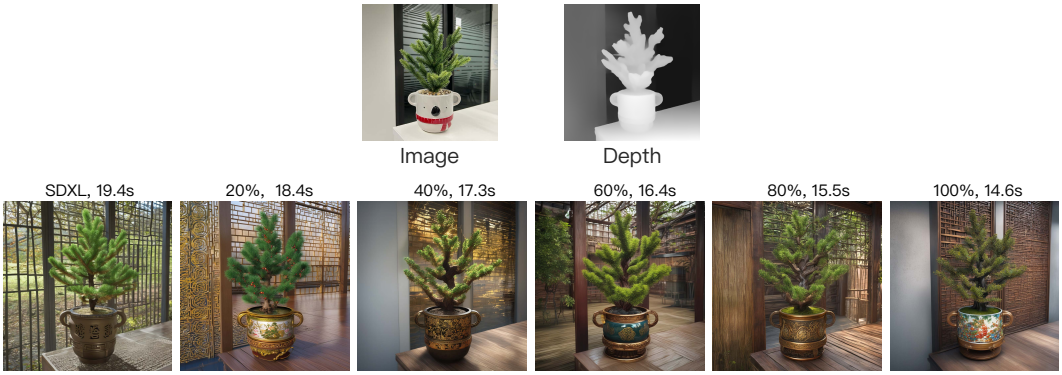
Negative Prompt: sloppy, messy, blurry, noisy, highly detailed, ultra textured, photo, realistic

Figure 29: T-Stitch with more complex prompts based on SDXL (Podell et al., 2023) and SSD-1B (Segmind, 2023). We annotate the faction of SSD-1B on top of images. Time cost is measured by generating one image on RTX 3090.

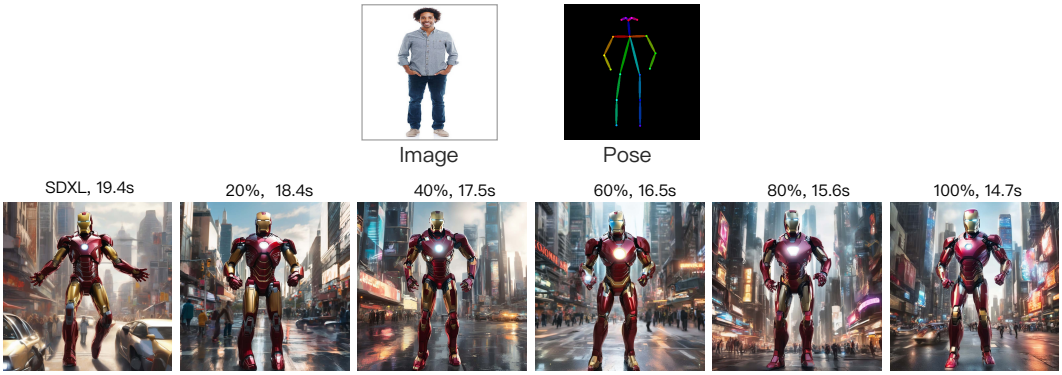
The optimal threshold for T-Stitch. In our observation, stitching at the early 40% minorly affects the generation quality, while at the larger fractions, T-Stitch provides a clear trade-off between



Prompt: highly detailed cinematic aerial view of a futuristic research complex in Antarctica, 4K
Negative Prompt: blur, low quality, bad quality, sketches



Prompt: a beautiful bosai tree, masterpiece, 4K
Negative Prompt: blur, low quality, bad quality, sketches



Prompt: Ironman dancing in a futuristic city, high quality
Negative Prompt: low quality, bad quality

Figure 30: T-Stitch with SDXL-based ControlNet. We annotate the faction of SSD-1B on top of images. Time cost is measured by generating one image on one RTX 3090.

a small and large model. This phenomenon has been observed across various architectures and samplers. However, note that this 40% threshold might not hold for all use cases.

In Figure 36, we show that different classifier-free guidance (CFG) scales may affect this optimal threshold during our experiments with DiT models. By default, we set the CFG scale to 1.5 as it is the default value in DiT evaluation. However, under the CFG scale of 2.0, we observe this optimal

Figure 31: Based on Stable Diffusion v1.4 and BK-SDM Tiny, we generate images by different fractions of BK-SDM for **8 consecutive runs** (a for-loop) on one GPU. T-Stitch demonstrates stable performance for robust image generation. Best viewed in digital version and zoom in.

Table 14: Summary of FID-5K and acceleration results for various configurations in Figure 1.

Name	**FID-5K**	**Acceleration**
DiT-XL (our implementation)	9.20	-
T-Stitch (10%)	9.17	7.84%
T-Stitch (20%)	8.99	18.71%
T-Stitch (30%)	9.03	32.00%
T-Stitch (40%)	9.95	50.00%
T-Stitch (50%)	10.06	75.53%

threshold occurs at around 60%. But we should not assume a larger CFG scale would always help T-Stitch since FID sometimes cannot reflect the desired image quality, as mentioned in the SDXL report. We aim to demonstrate that the optimal threshold could be affected by CFG scale, as one of the limitations of T-Stitch.

It is also intuitive that different models behave differently when using T-Stitch. For example, a) In Figure 37, the best cut-off for DiT at CFG scale of 1.5 is around 40%. b) On the other hand, the experiment on BK-SDM Tiny and SD v1.4 indicates that the optimal cut-off exists before the 40% estimate. c) In Table 1, we have already shown that LDM-S can replace 50% early steps for the baseline LDM with comparable or even better FID.

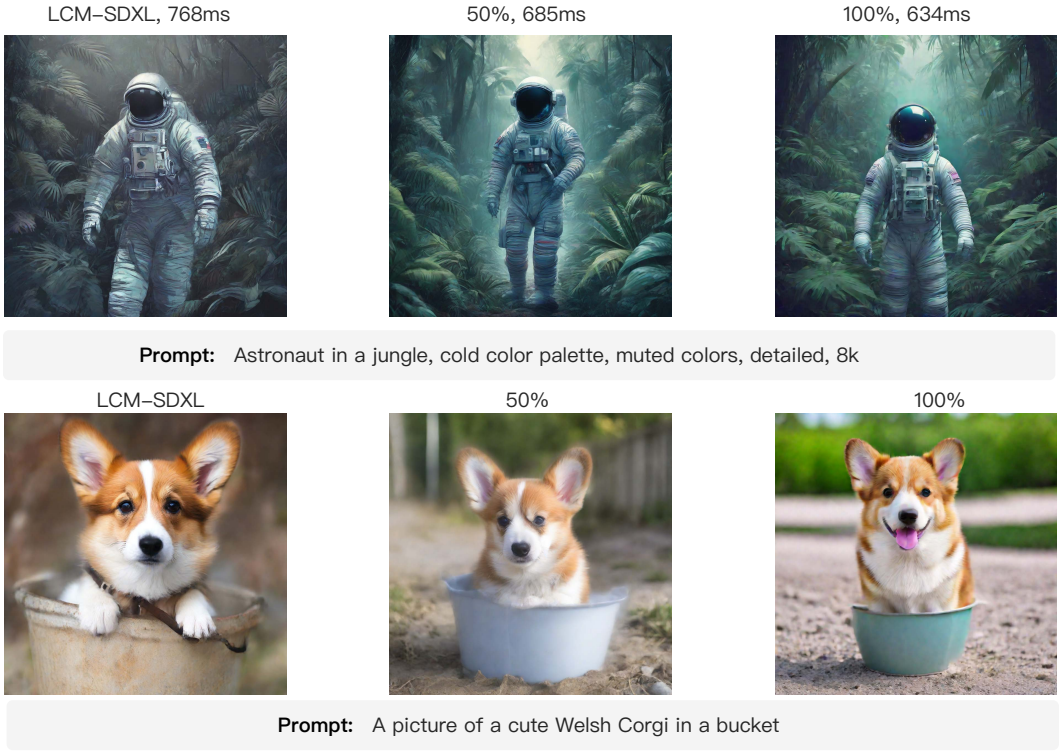


Figure 32: T-Stitch based on distilled models: LCM-SDXL (Luo et al., 2023) and LCM-SSD-1B (Luo et al., 2023), under **2 sampling steps**. We annotate the faction of LCM-SSD-1B on top of images. Time cost is measured by generating one image on RTX 3090 in milliseconds.

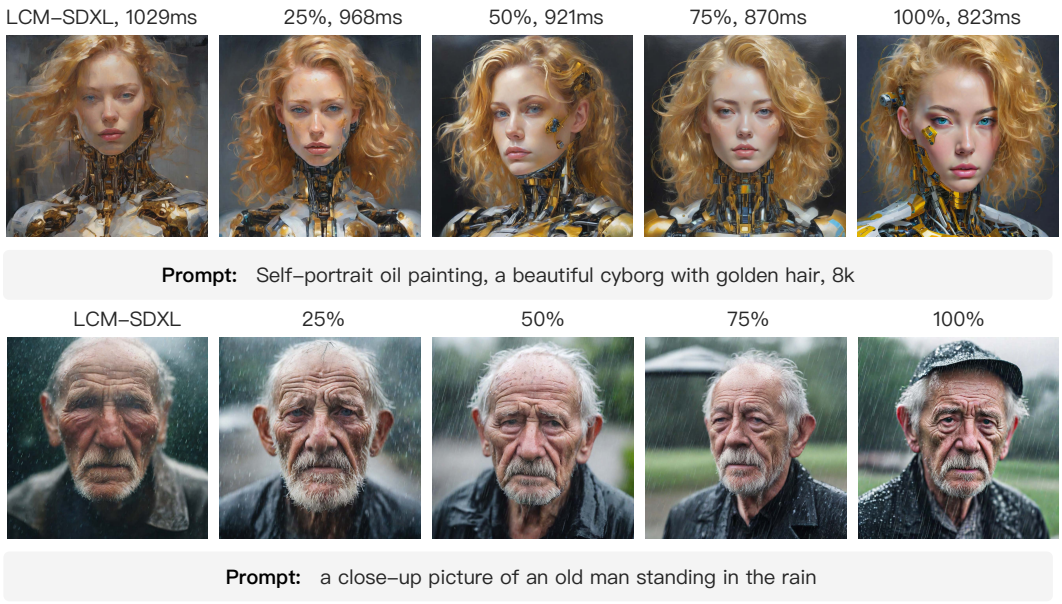


Figure 33: T-Stitch based on distilled models: LCM-SDXL (Luo et al., 2023) and LCM-SSD-1B (Luo et al., 2023), under **4 sampling steps**. We annotate the faction of LCM-SSD-1B on top of images. Time cost is measured by generating one image on RTX 3090 in milliseconds.

At this stage, determining the optimal switching point in T-Stitch can be done very efficiently, as discussed in our general response. For optimal results, we recommend conducting these efficient

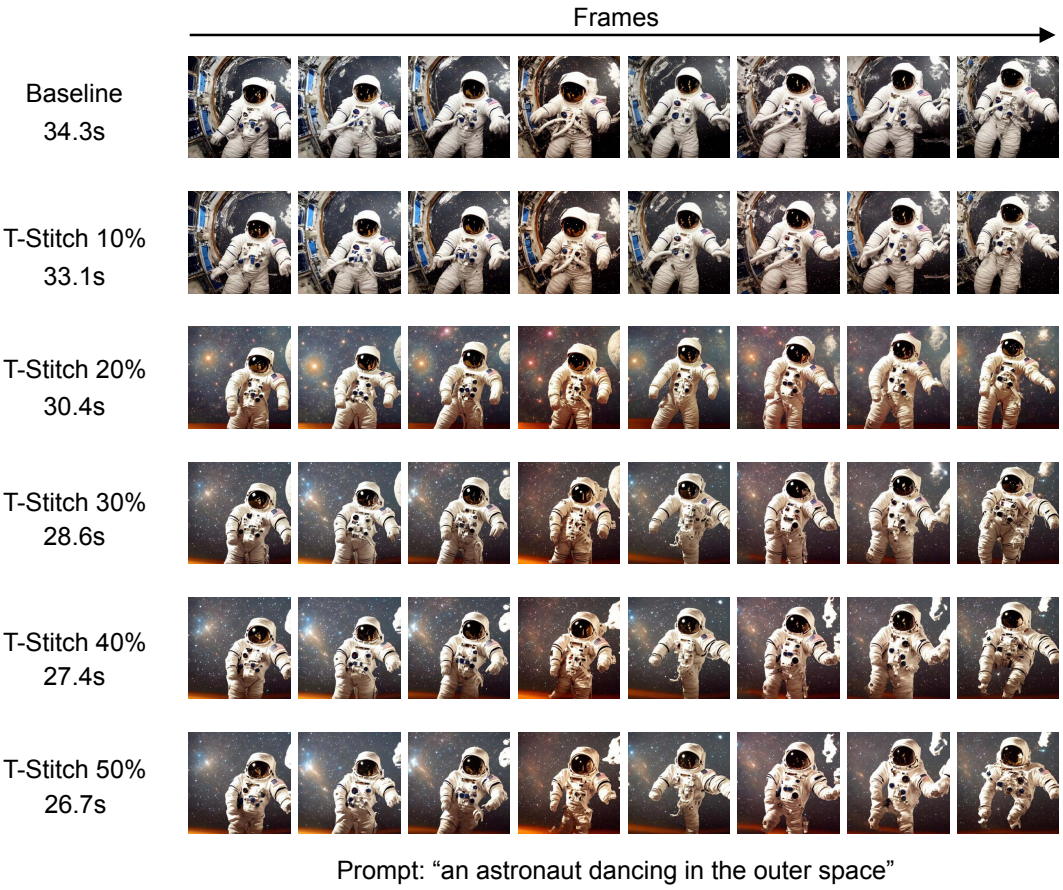


Figure 34: T-Stitch based on Text2Video-Zero Khachatryan et al. (2023).

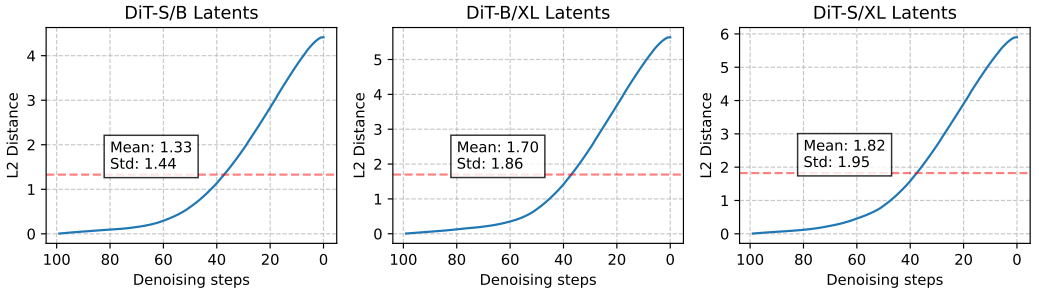


Figure 35: L2-distance comparison of latent embeddings at different denoising steps between DiT models. Results are averaged over 32 images.

preliminary experiments to determine the ideal switching point for specific model pairs. We leave more interesting explorations in future works.

T-Stitch beyond pairwise. The requirement of pretrained models naturally becomes one limitation for T-Stitch beyond pairwise since it relies on publicly available model weights. Furthermore, different models may have different optimal CFG scales. This means that combining multiple models along the same sampling trajectory creates a much larger space compared to pairwise combinations, making comprehensive evaluation challenging.

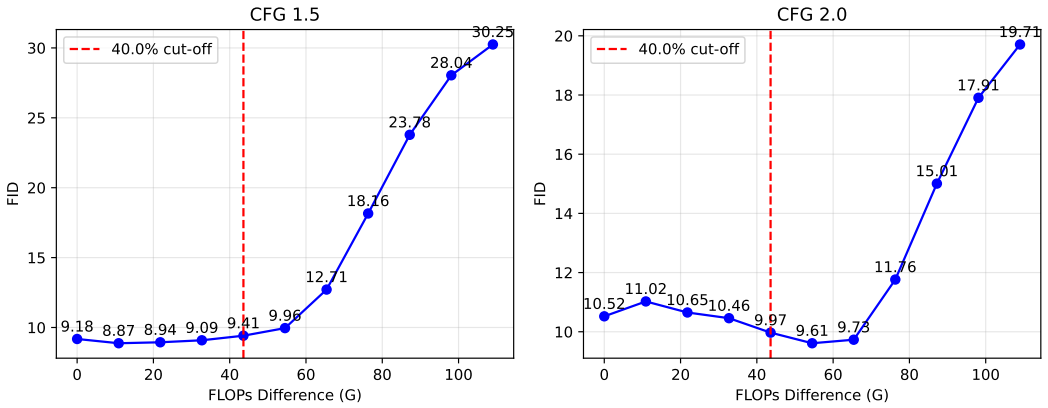


Figure 36: FLOPs difference versus FID. We report the DiT-S/XL comparison based on classifier-free guidance scale of 1.5 and 2.0 under 50 timesteps and DDIM sampler.

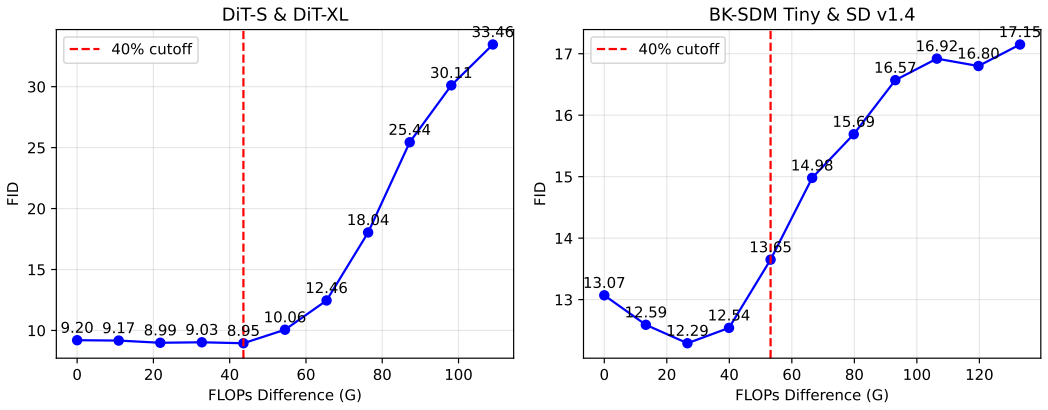


Figure 37: FLOPs difference versus FID. We report the DiT-S/XL comparison based on ImageNet-1K and 100 timesteps. The experiments on BK-SDM Tiny and SD v1.4 are based on MS-COCO benchmark and 50 timesteps.