Learning Few-Step Diffusion Models by Trajectory Distribution Matching

Supplementary Material

A. Training Algorithm

We present the algorithm for distilling K-step TDM in Algorithm 1.

Algorithm 1 Trajectory Distribution Matching.

Require: learning rate η, desired sampling steps K, total iterations N, real score f_ϕ.
Ensure: optimized models f_θ, f_ψ.

```
Initialize weights {θ, ψ} by φ;
for i ← 1 to N do
    Sample noise ϵ from standard normal distribution;
    Sample {x_{t_i}}_{i=0}^{K-1} with initialized noise ϵ from generator f_θ by K steps via ODESolver.
    Sample x_{t_m} from {x_{t_i}}_{i=0}^{K-1}.
    Sample Timesteps t from t_m to t_{m+1}.
    Obtain noisy samples x_t ~ q(x_t|x_{t_m})
    # update fake score
    Compute Loss L_ψ following Line 234.
    ψ ← ψ − η∇_ψ L_ψ;
    # update Generator
    Compute Loss L_θ following Eq. (11).
    θ ← θ − η∇_θ L_θ;
end for
```

B. Derivations

Derivations of Eq. (9). Given the case without noise, the learning objective of the fake score s_ψ by score matching is:

$$\begin{aligned} L(\psi) =& \mathbb{E}_{p_{K_1}(\mathbf{x})} ||\nabla \log p_{K_1}(\mathbf{x}) - s_\psi(\mathbf{x})||_2^2 \\ &+ \mathbb{E}_{p_{K_2}(\mathbf{x})} ||\nabla \log p_{K_2}(\mathbf{x}) - s_\psi(\mathbf{x})||_2^2. \end{aligned} \tag{13}$$

Its gradient can be computed as follows:

$$\begin{aligned} \nabla_\psi L(\psi) = \int [&2p_{K_1}(\mathbf{x})(\nabla \log p_{K_1}(\mathbf{x}) - s_\psi(\mathbf{x})) \\ &+2p_{K_2}(\mathbf{x})(\nabla \log p_{K_2}(\mathbf{x}) - s_\psi(\mathbf{x}))] \frac{\partial s_\psi(\mathbf{x})}{\partial \psi} d\mathbf{x} \end{aligned} \tag{14}$$

The global minimum is achieved when $\nabla_\psi L(\psi) = \mathbf{0}$. It is clear that when

$$s_\psi(\mathbf{x}) = \frac{p_{K_1}(\mathbf{x})\nabla_\mathbf{x} \log p_{K_1}(\mathbf{x}) + p_{K_2}(\mathbf{x})\nabla_\mathbf{x} \log p_{K_2}(\mathbf{x})}{p_{K_1}(\mathbf{x}) + p_{K_2}(\mathbf{x})}, \tag{15}$$

we have $\nabla_\psi L(\psi) = \mathbf{0}$. Hence the optimal fake score is given in Eq. (15).

Derivations of Eq. (6). The gradient of Eq. (6) can be computed as following:

$$\begin{aligned} \frac{\partial L(\theta)}{\partial \theta} &= \sum_{i=0}^{K-1} \sum_{\tau=t_i}^{t_{i+1}} \frac{\lambda_\tau}{2} \frac{\partial ||\mathbf{x}_{t_i} - \text{sg}(\tilde{\mathbf{x}}_{t_i})||_2^2}{\partial \mathbf{x}_{t_i}} \frac{\partial \mathbf{x}_{t_i}}{\partial \theta} \\ &= \sum_{i=0}^{K-1} \sum_{\tau=t_i}^{t_{i+1}} \frac{\lambda_\tau}{2} [2(\mathbf{x} - \text{sg}(\tilde{\mathbf{x}}_{t_i}))] \frac{\partial \mathbf{x}_{t_i}}{\partial \theta} \\ &= \sum_{i=0}^{K-1} \sum_{\tau=t_i}^{t_{i+1}} \lambda_\tau [s_\psi(\mathbf{x}_\tau, \tau) - s_\phi(\mathbf{x}_\tau, \tau)] \frac{\partial \mathbf{x}_{t_i}}{\partial \theta}. \end{aligned} \tag{16}$$

This is the same as the approximated gradient of the original objective:

$$\begin{aligned} \nabla_\theta L(\theta) &= \sum_{i=0}^{K-1} \sum_{\tau=t_i}^{t_{i+1}} \lambda_\tau [\nabla_{\mathbf{x}_\tau} \log p_{\theta,\tau|t_i}(\mathbf{x}_\tau) - s_\phi(\mathbf{x}_\tau, \tau)] \frac{\partial \mathbf{x}_{t_i}}{\partial \theta} \\ &\approx \sum_{i=0}^{K-1} \sum_{\tau=t_i}^{t_{i+1}} \lambda_\tau [s_\psi(\mathbf{x}_\tau, \tau) - s_\phi(\mathbf{x}_\tau, \tau)] \frac{\partial \mathbf{x}_{t_i}}{\partial \theta} \end{aligned}$$

C. Additional Related Works

Recently, there mainly have been two lines in diffusion distillation: Trajectory Distillation and Distribution Matching. Trajectory distillation tries to distill a few-step student model by simulating the generative process of DMs. These methods typically predict the multi-step solution of PF-ODE solver by one step [12, 21, 27, 39]. Consistency family [10, 17, 33, 35, 43] enforces self-consistency. These methods suffer from numerical errors when solving pre-trained PF-ODE. Distribution Matching tries to distill student models via match at the distribution level. Diffusion-GAN hybrid models [9, 29, 38] have been proposed for this aim, however, stabilizing GAN training requires real data, careful architecture design, and auxiliary regression losses. Another promising way for distribution matching is through score distillation [19, 22, 36, 41, 44]. These methods typically ignore the intermediate steps of the trajectory. Our method explores trajectory distillation at the distribution level, enjoying the best of two worlds. Although Hyper-SD [25] explores combining trajectory distillation and distribution matching, their works treat these two techniques as distinguished parts, requiring multiple training objectives and multiple training stages. In contrast, our proposed objective naturally unifies trajectory distillation and distribution matching, providing a highly efficient and effective distillation method. Besides, motivated by the similarity between consistency models and our proposed

method in learning generator, we propose a surrogate training objective, introducing the Huber metric into training. This leads to better performance and potentially encourages the community to explore other types of distance metrics in distilling via distribution matching. Recently, DMD2 [42] also explored distilling a few-step generator via distribution matching. However, our method is fundamentally different from their work. Their work tries to predict clean images at different timesteps, ignoring alignment with the teacher's trajectory. This leads to harder learning and slower convergence. In particular, in distilling 4-step SDXL, we only require 1.25% of the training cost for DMD2, while achieving significantly better performance. Besides, a recent work MMD [28] also developed a multi-step generator based on a similar style with score distillation. However, our work is essentially different from them, since MMD applies moment matching for diffusion distillation, while we propose a new distillation paradigm that unifies trajectory distillation and distribution matching. Specifically, MMD employs moment matching to train both generator and fake "score" which is fundamentally different from score matching as noted in their paper [28]; In contrast, we use reverse KL to train the generator and score matching to train the fake score. Moreover, MMD uses ancestral sampling (DDPM sampler [7]) from noisy real data in training, which introduces large stochasticity in intermediate samples, is sub-optimal for few-step sampling. In contrast, we use deterministic sampling from noise which is image-free, more effective for fewer-step sampling, and builds a non-trivial connection with trajectory distillation.

Additionally, a concurrent work [3] explored flexible deterministic sampling but was limited by point-to-point self-consistency and required training the generator at arbitrary timesteps, challenging model capacity. Its extension to text-to-image generation also remains unclear. In contrast, our method trains the generator at only K timesteps with distribution-level matching, enabling more effective flexible deterministic sampling. We further achieve state-of-the-art performance in text-to-image generation.

D. Experiment details

We use the AdamW optimizer for both the generator and fake score. By default, the β_1 is set to be 0, the β_2 is set to be 0.999.

SD-v1.5 We adopt a constant learning rate of 2e-6 for the generator and 2e-5 for the fake score. We apply gradient norm clipping with a value of 1.0 for both the generator and fake score. We use batch size 256. We set the CFG as 3.5. Generally, the training is done within 20k iterations for unified training and within 3k iterations for specific 4-step training.

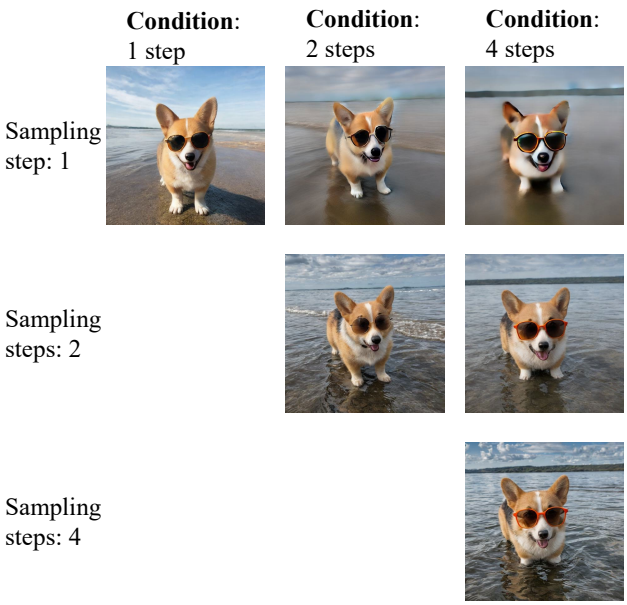


Figure 8. Visual samples of varying the condition steps and sampling steps. The prompt is "A corgi with sunglasses, traveling in the sea""

SDXL We adopt a constant learning rate of 1e-6 for the generator and 5e-6 for the fake score. We apply gradient norm clipping with a value of 1.0 for both the generator and fake score. We use batch size 64. We set the CFG as 8. Since SDXL has 2.7B parameters, fine-tuning it at 1024 resolution is computationally expensive. We first fine-tune for 1k iterations at 512 resolution, then fine-tune for another 1k iterations at 1024 resolution. The fake score is initialized from the pre-trained SDXL in both stages.

PixArt-α We adopt a constant learning rate of 2e-6 for the generator and 2e-5 for the fake score. We set the CFG as 3.5. We apply gradient norm clipping with a value of 1.0 for both the generator and fake score. We use batch size 32. The training can be done within 500 iterations.

E. Additional Experiments

The Effect of Varying Sampling Steps. We investigate the impact of different sampling and conditioning configurations, with results visualized in Fig. 8. The results show that TDM-unify exhibits systematic behavioral variations across different combinations of conditional and sampling steps. Notably, the model demonstrates an intrinsic understanding of the underlying ODE trajectory, adaptively positioning itself at appropriate points along this path given the specified condition steps.

Comparison on Diversity We quantitatively and visually compared our method with the most direct baseline

Table 6. Additional comparison to the most direct baseline and base model. TDM on SD 1.5 here is TDM-unify-GAN in Tab. 1, which is initialized from original SD 1.5 and adopts the same GAN loss as DMD2. The same CFG is applied across methods.

Method	SD 1.5			SDXL		
	HPS↑	Diversity Score↑	Impr. Recall↑	HPS↑	Diversity Score↑	Impr. Recall↑
Base Model	25.50	0.66	0.68	33.19	0.65	0.76
DMD2	29.49	0.61	0.54	31.46	0.61	0.70
TDM	**30.83**	**0.64**	**0.59**	**34.88**	**0.63**	**0.73**

Table 7. Comparison on HPS across variants in 4-step generation based on SD-v1.5.

Train-DDIM	Train-DPMSolver	Test-DDIM	Test-DPMSolver	HPS↑
✓		✓		31.04
✓			✓	31.35
	✓	✓		30.86
	✓		✓	31.30

Table 8. Comparison on HPS across variants in 4-step generation based on SD-v1.5.

Method	HPS↑
TDM (Matching noisy samples $\mathbf{x}_{t_i}$)	31.35
Matching clean samples $\hat{\mathbf{x}}_{t_i}$	24.63

Table 9. The effect of using more expensive Fisher Divergence in 4-step generation.

Method	HPS↑
TDM	31.35
TDM w/ Fisher	**31.70**

DMD2 and the base model. We adopt Improved recall and diversity score as metrics. The results in Tab. 6 and Fig. 9 indicate that our method is more diverse than DMD2.

F. Ablation Studies

F.1. Ablation Details

We use the same hyperparameters and training iterations for all variants, with differences only in the ablating components.

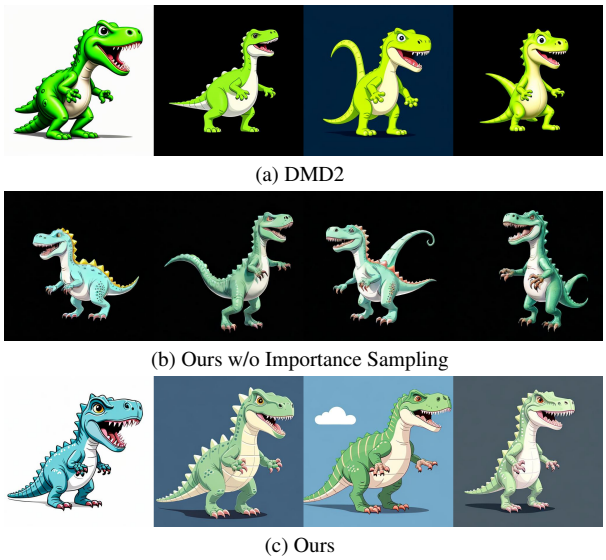
The formulation of using GANs during distillation Following the previous work [11, 20], we use latent discriminators, with the backbone based on the UNet encoder from SD-v1.5. In particular, we perform the following loss for learning generator:

$$\mathbb{E}_K \sum_{i=0}^{K-1} \sum_{\tau=t_i^K}^{t_{i+1}^K} \{\lambda_\tau \mathrm{KL}(p_{\theta,\tau|t_i^K}(\mathbf{x}_\tau|K)||p_{\phi,\tau}(\mathbf{x}_\tau)) + \lambda_\tau \mathbb{E}_{p_{\theta,\tau|t_i^K}(\mathbf{x}_\tau|K)} \mathrm{ADVLoss}(\mathbf{x}_\tau, \tau, K)\}. \tag{17}$$

The ADVLoss is the adversarial loss. Note that for a fair comparison, we inject the desired sampling steps K into GAN's discriminator too.

Details in implementing original loss We use normalization proposed in DMD [41], while we do not use the normalization in our proposed surrogate loss.

Details in implementing DMD2 Following the original DMD2 paper [42], we update the fake score 10 times per iteration. Other hyper-parameters remain consistent with our configuration.



(a) DMD2

(b) Ours w/o Importance Sampling

(c) Ours

Figure 9. Comparison on Mode Cover in 4-step generation based on SD-v1.5. It is clear that our method has better mode cover and image quality. The prompt is "A cute dinosaur, cartoon style"

F.2. Additional Ablation

To gain a more comprehensive understanding of our proposed methods, We conducted additional ablation studies in this section.

(a) matching clean samples

(b) Ours (matching noisy samples)

Figure 10. Comparison on the compatibility with deterministic samplers in the 4-step generation on SD-v1.5. It is clear that our method (matching noisy samples) has better visual quality.

Comparison on Mode Coverage. We found that using importance sampling for learning the fake score led to improved performance (Tab. 5) and better mode coverage. As shown in Fig. 9, our method demonstrates notably superior image quality and mode coverage. This improvement may be attributed to the fact that the fake score cannot accurately track the student distribution without using importance sampling. We additionally compare to the concurrent work DMD2 [42]. We found that DMD2's generated results also suffer from mode collapse, while its impact is somewhat less severe compared to the variant without importance sampling. This may be due to DMD2 trains the fake score multiple times at each iteration, resulting in a more accurate fake score at the cost of slow training.

Effect of Different ODE Solvers. In the experiments presented in the main body, we use DDIM [32] as the ODE solver during training and DPMsolver [15] during inference. Here we ablate the choice of ODE solver, with results shown in Tab. 7. We find that using DDIM versus DPM during training yields similar performance, which may be attributed to two reasons: 1) regardless of whether DDIM or DPM is used during training, we can utilize DPM sampling at test time to improve performance; 2) our training only backpropagates through one ODE step, preventing higher-order ODE solvers like DPMSolver from benefiting from higher-order information in the training.

Matching noisy samples $\mathbf{x}_{t_i}$ v.s. clean samples $\hat{\mathbf{x}}_{t_i}$ A core design of our method is to align noisy samples $\mathbf{x}_{t_i}$ predicted by the model with the target diffusion, rather than the clean samples $\hat{\mathbf{x}}_{t_i}$ predicted by the model. This design makes the support of deterministic sampling possible. We conduct experiments on matching clean samples, the results are shown in Tab. 8 and Fig. 10. It is clear that our method has a better performance, while matching clean samples deliver a poor deterministic sampling with notable artifacts.

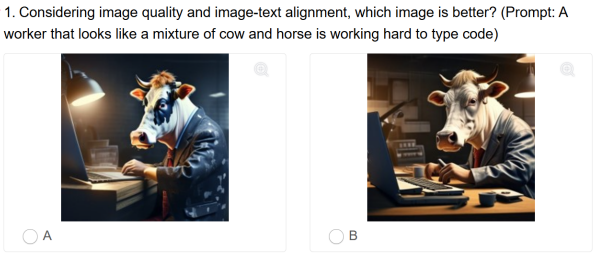


Figure 11. An example of the evaluation question for our user study.

Flexibility for using different distribution divergence The proposed TDM has the flexibility for using different distribution divergence instead of reverse KL divergence. In particular, the performance of TDM can be further improved by more expensive Fisher divergence (Tab. 9).

G. User Study Details

We conducted user research by presenting users with two anonymous images generated by different models and asking them to select the sample with higher image quality and better prompt alignment. We randomly selected 20 prompts for image generation. Each image was manually verified to ensure the absence of inappropriate or dangerous content. An example of an evaluation question is shown in Fig. 11. In total, we collected approximately 40 user responses.

H. Additional Qualitative Results

We present the additional visualization of ODE trajectory with clean samples at different timesteps in Fig. 14.

We present the visual samples of interesting LoRA into unseen customized models in Figs. 12 and 13. It can be seen that compared to the competing baseline, our method shows better visual quality and better style preservation.

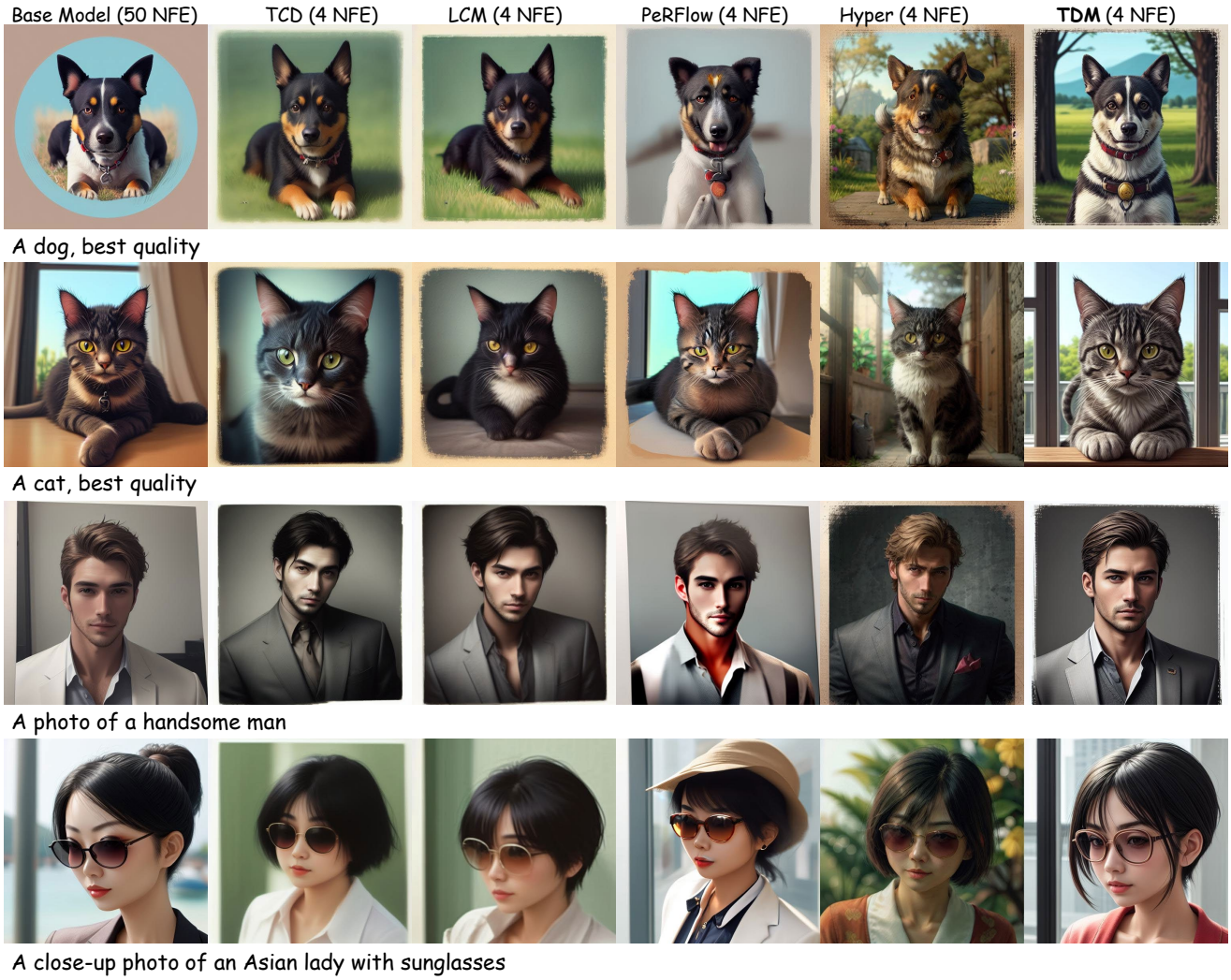


Figure 12. Additional Samples of integrating LoRA into unseen customized models - dreamshaper.

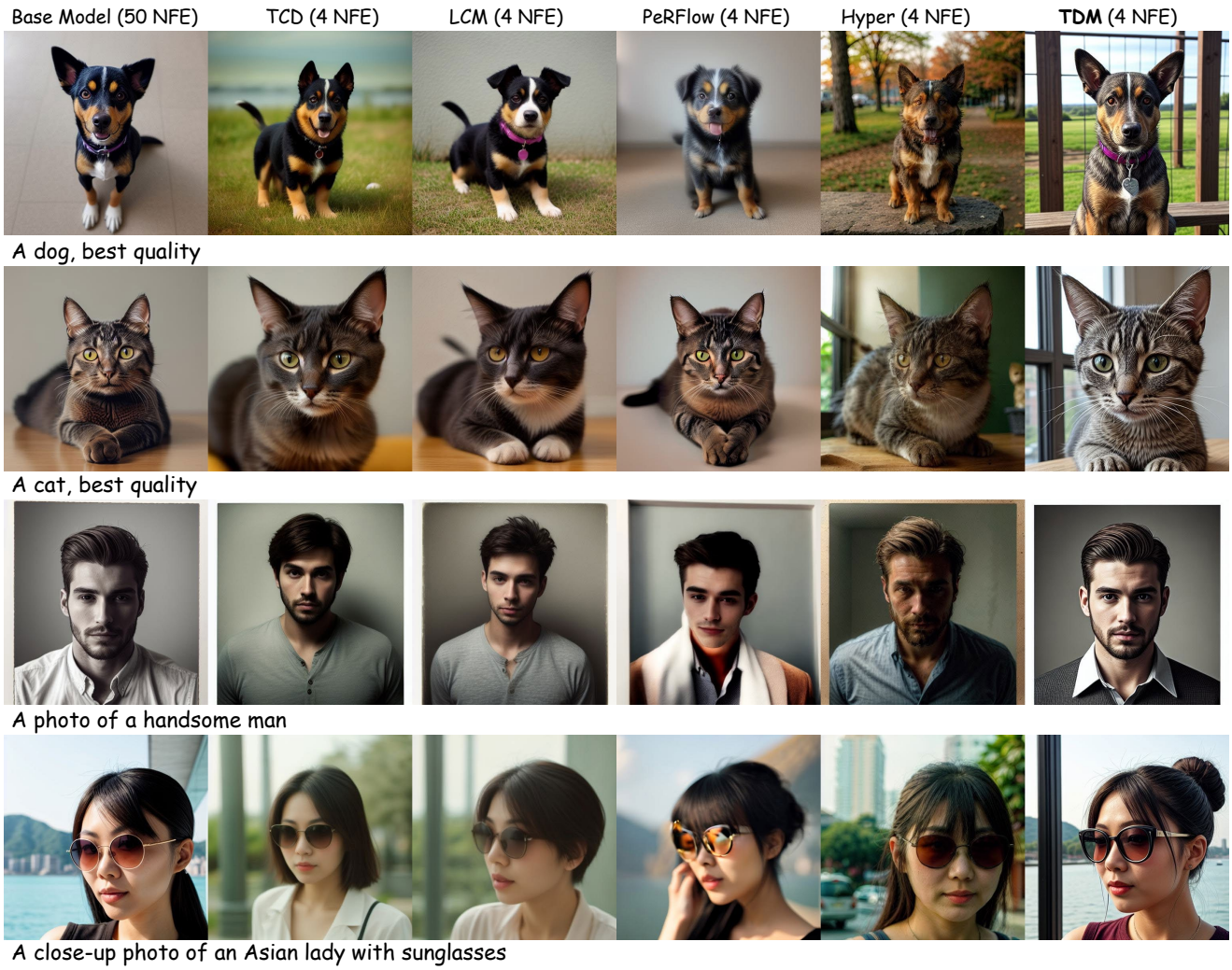


Figure 13. Additional Samples of integrating LoRA into unseen customized models - realisticvision.

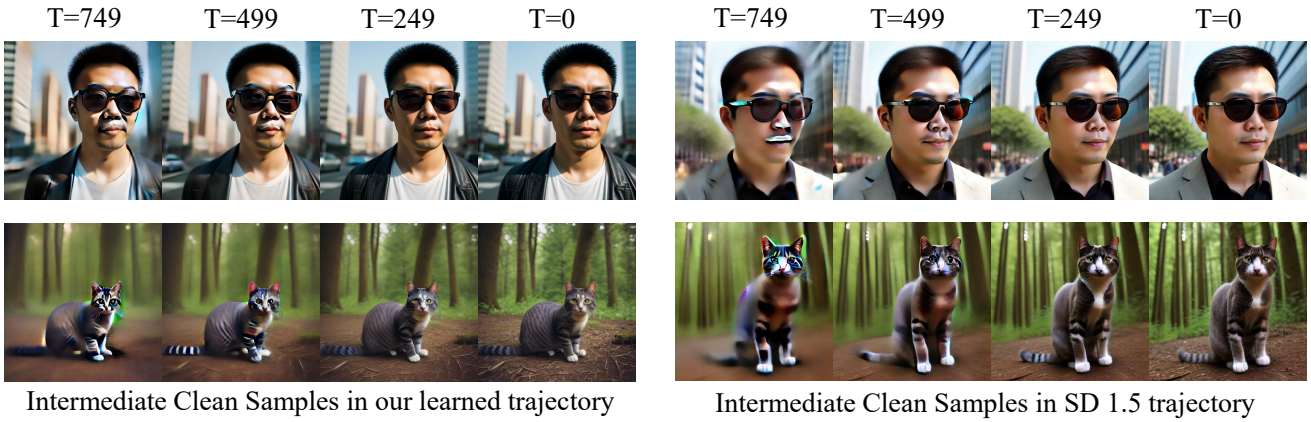


Figure 14. Additional visualization of ODE trajectory with clean samples at different timesteps. It is clear that our method suffers less from the CFG artifact and has better visual quality.