ONE STEP GENERATION VIA CONDITION SHIFTING

Anonymous authors
Paper under double-blind review

ABSTRACT

One-step text-to-image generation must preserve image quality and text alignment without repeated sampling updates. Trajectory-based objectives learn the mapping from noise to data, while distribution matching provides feedback on the samples that the generator actually produces. This feedback often requires an auxiliary network to estimate the generated distribution, increasing training cost as the backbone grows. Self-adversarial methods reduce this cost by learning real and fake trajectories within one model, using separate inputs for each task. We propose **APEX**, which learns the fake trajectory through condition shifting while retaining the original positive-time domain. Under the original condition, the model learns the real trajectory; under a deterministically shifted condition, it learns from its own outputs for the same text prompt. Flow matching on generated samples trains the shifted-condition prediction to follow the current generated distribution. Evaluating both velocities at the same noisy sample and time gives an internal distribution-matching correction, without a separate discriminator or fake-score network. Under score–velocity duality, this difference estimates the score discrepancy between the reference and generated distributions. We use it to construct a detached target for the one-step prediction and combine the resulting correction with the RCGM base objective. Both tasks share the existing network parameters, and inference uses only the original condition in a single forward pass. Across 0.6B–20B backbones, APEX supports full-parameter and LoRA tuning. SANA 0.6B achieves 0.84 GenEval with 0.20-second latency at one network function evaluation (NFE), and Qwen-Image 20B with LoRA reaches 0.89 GenEval at 1 NFE. Under matched training on Z-Image 6B at 2 NFEs, APEX improves over TwinFlow on all five reported metrics, including Chinese and English text rendering by 10.03 and 14.09 points. Condition ablations favor affine shifting over the other tested constructions on GenEval, DPG-Bench, and WISE.



Figure 1: **Examples of text-to-image generation with APEX.**

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Diffusion and flow matching models produce high-quality, semantically aligned images by progressively transforming noise into data (Ho et al., 2020; Lipman et al., 2022). Their repeated network evaluations make inference costly, motivating methods that generate images in fewer steps (Song et al., 2023; Salimans & Ho, 2022). At one step, a single prediction must preserve image structure, visual detail, and the relationships specified by the text prompt. Consistency and trajectory-based objectives learn this mapping, while distribution matching (Yin et al., 2024b; Zheng et al., 2025b) provides additional feedback on the generated samples during training.

Distribution Matching Distillation (DMD) compares the scores of the reference and current generated distributions (Yin et al., 2024b). The reference score comes from a pretrained model, while an auxiliary fake-score network learns from the generator's outputs. The score difference tells the generator how its current outputs differ from the reference distribution. During fine-tuning, the generated distribution changes with the model parameters, so the fake-score estimate must be updated using new generated samples. The pretrained reference provides the target score, but cannot supply this changing estimate. Maintaining the auxiliary network increases both training cost and memory use, especially when it has the same scale as the generator. Learning the fake trajectory within the generator offers a way to obtain this feedback without maintaining another network.

TwinFlow (Cheng et al., 2025) learns real and fake trajectories within a shared model and uses their velocity discrepancy for self-adversarial correction. Positive and negative time inputs select the real and fake trajectories, respectively, allowing the same network to estimate both velocities. For conditional generation, the conditioning input offers another way to distinguish these tasks while keeping both trajectories in the original positive-time domain. For a text-conditioned model, the same prompt must remain associated with both trajectories so that their discrepancy compares the reference and generated distributions for that prompt under the same noising process.

We introduce **APEX**, which constructs the fake trajectory through condition shifting. Under the original text condition, the model learns from reference data. Under the shifted condition, it learns through flow matching on its own generated samples for the same prompt. We obtain this condition by applying a deterministic affine transformation to the text embedding. An invertible transformation preserves the prompt information. The generated samples retain their association with the original prompt, while the shifted embedding distinguishes their supervision from that of the reference data. This allows the model to learn both trajectories using the same architecture and time domain. The real and fake velocities are evaluated at the same noisy sample and time, and their difference supplies a detached correction for the one-step prediction under the original condition.

APEX follows the internal self-adversarial setting of TwinFlow and the score-difference supervision of DMD. Its contribution is the condition-space construction of the fake branch and its evaluation under matched training. We combine the correction with the RCGM base objective (Sun & Lin, 2025), which learns generative mappings across time intervals. At inference, only the original condition is used, so the fake branch adds no sampling evaluations.

Our evaluations span 0.6B–20B backbones with full-parameter and LoRA tuning. SANA 0.6B reaches 0.84 GenEval at 1 NFE with 0.20-second latency, and Qwen-Image 20B with LoRA reaches 0.89 GenEval. Matched retraining on Z-Image 6B gives higher scores than TwinFlow on all five metrics at 2 NFEs, with the largest gains in Chinese and English text rendering: 10.03 and 14.09 points, respectively. Matched Qwen-Image retrains improve GenEval and WISE at both sampling budgets, with slightly lower DPG-Bench scores than TwinFlow. The matched studies hold the backbone, data, tuning mode, and update budget fixed, while the system-level results describe quality and inference cost across model scales. Condition ablations compare affine shifting with a discrete fake token, null conditioning, and Gaussian perturbation. Affine shifting gives the highest scores among these constructions; the parameter study finds that increasing the bias can either improve or reduce GenEval, depending on the scale.

Our main contributions are:

a. **Condition shifting for self-adversarial learning.** We distinguish real and fake trajectories through the condition input of a shared model, retaining the original positive-time domain.
b. **Internal velocity-matching correction.** Building on DMD and TwinFlow, we use these trajectories to correct one-step predictions alongside RCGM training.
c. **Evaluation across scales and tuning modes.** Matched TwinFlow comparisons and condition ablations assess the construction, with full-parameter and LoRA results from 0.6B to 20B. These evaluations cover compositional alignment, prompt following, and world knowledge.

2 PRELIMINARIES

We review flow matching, few-step generation, and DMD; $\mathbf{c}$ denotes the text embedding.

Continuous Generative Models. Conditional diffusion (Ho et al., 2020; Song et al., 2020) and flow matching (Lipman et al., 2022) transport Gaussian noise $\mathbf{z} \sim p_z = \mathcal{N}(\mathbf{0}, \mathbf{I})$ to data $\mathbf{x} \sim p_r(\cdot \mid \mathbf{c})$. For $t \in [0, 1]$, we define the noising path between each data sample and its Gaussian noise as

$$\mathbf{x}_t = \alpha(t)\mathbf{z} + \gamma(t)\mathbf{x}\,. \tag{1}$$

We use the optimal transport (OT) path with $\alpha(t) = t$ and $\gamma(t) = 1 - t$, so $\mathbf{x}_1 = \mathbf{z}$ and $\mathbf{x}_0 = \mathbf{x}$. The velocity defines the probability-flow ODE (PF-ODE):

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \mathbf{v}(\mathbf{x}_t, t, \mathbf{c})\,. \tag{2}$$

Flow matching learns this field by regressing to $\mathbf{u}_r(\mathbf{x}, \mathbf{z}) := \mathbf{z} - \mathbf{x}$:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{\mathbf{x},\,\mathbf{c},\,\mathbf{z},\,t}\left[\|\boldsymbol{F}_\theta(\mathbf{x}_t, t, \mathbf{c}) - \mathbf{u}_r(\mathbf{x}, \mathbf{z})\|^2\right]\,. \tag{3}$$

The expectation uses paired data $(\mathbf{x}, \mathbf{c})$, independent $\mathbf{z} \sim p_z$, and $t \sim \mathcal{U}(0, 1)$. The population minimizer is the conditional mean velocity $\mathbf{v}_r^*(\mathbf{x}_t, t, \mathbf{c}) = \mathbb{E}[\mathbf{z} - \mathbf{x} \mid \mathbf{x}_t, t, \mathbf{c}]$. Sampling integrates from $t = 1$ to $t = 0$, for example with Euler's method (Karras et al., 2022).

Few-Step Generation. Consistency Models map points on an ODE trajectory to a common endpoint (Song et al., 2023). RCGM (Sun & Lin, 2025) and MeanFlow (Geng et al., 2025a) learn interval mappings and average velocities for few-step sampling. We write the one-step map as $\boldsymbol{G}_\theta(\mathbf{z}, \mathbf{c}) = \mathbf{z} - \boldsymbol{F}_\theta(\mathbf{z}, 1, \mathbf{c})$. Here the prediction uses the full interval from 1 to 0; the velocity calls used for flow matching and score conversion are instantaneous. These calls share parameters but need not return the same velocity (Appendix B.1).

Distribution Matching. For a fixed condition, let p_θ be the distribution of $\boldsymbol{G}_\theta(\mathbf{z}, \mathbf{c})$ and p_r the reference distribution. Their noised marginals at time t have scores $\mathbf{s}_{i,t}(\mathbf{y}) := \nabla_{\mathbf{y}} \log p_{i,t}(\mathbf{y})$ for $i \in \{r, \theta\}$. DMD (Yin et al., 2024b) estimates the reverse-KL gradient using their score difference at the same noised generated sample. Its reference score is supplied by a pretrained teacher, while an auxiliary fake-score network is trained on current generator outputs. Thus, removing the auxiliary network still requires an estimate of the generated distribution. Under the OT path, score–velocity duality allows this difference to be obtained from the corresponding marginal velocities (Appendix B.2). Both fields are evaluated at the same noise level.

TwinFlow (Cheng et al., 2025) learns both velocities within one model, using positive time for the real trajectory and negative time for the fake trajectory. For a detached generated sample $\mathbf{x}^{\mathrm{fake}}$, fresh noise $\boldsymbol{\epsilon}' \sim p_z$ gives $\mathbf{x}_t^{\mathrm{fake}} = (1 - t)\mathbf{x}^{\mathrm{fake}} + t\boldsymbol{\epsilon}'$ at $t \in (0, 1)$. Writing $\boldsymbol{F}_\theta$ for its network, the fake-trajectory objective uses the corresponding flow-matching target:

$$\mathcal{L}_{\mathrm{adv}}^{\mathrm{TF}} = \mathbb{E}_{\mathbf{x}^{\mathrm{fake}},\,\boldsymbol{\epsilon}',\,t}\left[\|\boldsymbol{F}_\theta(\mathbf{x}_t^{\mathrm{fake}}, -t, \mathbf{c}) - (\boldsymbol{\epsilon}' - \mathbf{x}^{\mathrm{fake}})\|^2\right]\,. \tag{4}$$

APEX follows this internal construction and distinguishes the fake trajectory through the condition input. The corresponding score-difference expression for GAN updates is given in Appendix B.8 .

3 APEX

APEX learns real and fake trajectories in one network and uses their velocity discrepancy to correct one-step predictions under the original condition (Figure 2).

3.1 CONDITION SHIFTING FOR SELF-ADVERSARIAL TRAINING

The original condition $\mathbf{c}$ identifies the trajectory toward reference data $\mathbf{x} \sim p_r(\cdot \mid \mathbf{c})$. Condition shifting, $\mathbf{c}_{\mathrm{shift}} = a\mathbf{c} + b\mathbf{1}$, identifies the trajectory toward the generator's own samples for that same condition. We illustrate the construction with $\mathbf{c}_{\mathrm{shift}} = -\mathbf{c}$; the ablations report the scale and bias used in each comparison. The shifted condition selects the fake-trajectory task, whose targets still come from the original prompt. Both trajectories use $t \in (0, 1)$ and share the network parameters.

With independent $\mathbf{z}, \boldsymbol{\epsilon}' \sim p_z$ and $t' \sim \mathcal{U}(0, 1)$, generate a one-step sample and re-noise it. We write sg to indicate that the generated sample is detached:

$$\mathbf{x}_\theta^{\mathrm{sg}} := \mathrm{sg}(\mathbf{z} - \boldsymbol{F}_\theta(\mathbf{z}, 1, \mathbf{c}))\,, \qquad \mathbf{x}_{\theta,t'}^{\mathrm{sg}} := (1 - t')\mathbf{x}_\theta^{\mathrm{sg}} + t'\boldsymbol{\epsilon}'\,, \tag{5}$$

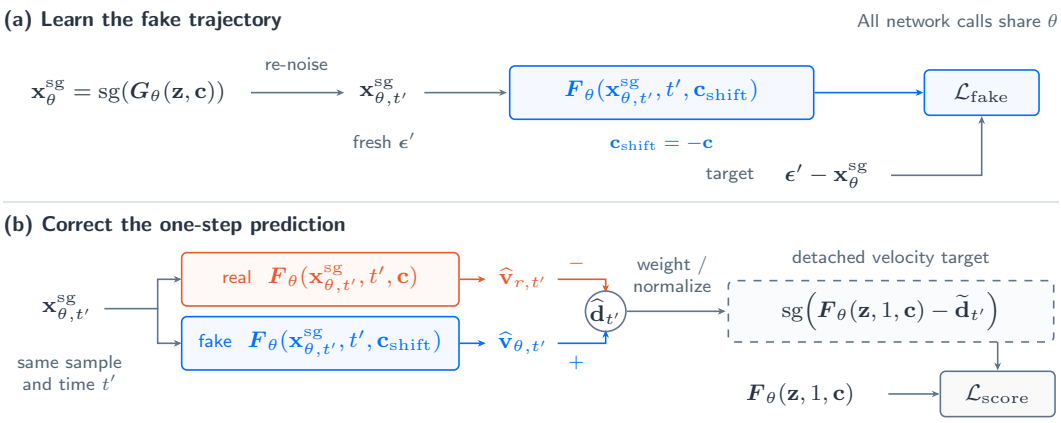

Figure 2: **APEX training overview**, illustrated with $\mathbf{c}_{\mathrm{shift}} = -\mathbf{c}$. (a) Generated samples supervise the fake trajectory. (b) The two velocities at the same noisy sample and time define a detached correction target for the one-step prediction. All calls share θ; the base objective is described in Section 3.3.

Flow matching under the shifted condition learns the fake velocity:

$$\mathcal{L}_{\mathrm{fake}}(\theta) := \mathbb{E}\left[\left\|\boldsymbol{F}_\theta(\mathbf{x}_{\theta,t'}^{\mathrm{sg}}, t', \mathbf{c}_{\mathrm{shift}}) - (\boldsymbol{\epsilon}' - \mathbf{x}_\theta^{\mathrm{sg}})\right\|_2^2\right]. \tag{6}$$

For a fixed generated distribution and an invertible condition transform, the population optimum is its conditional-mean velocity. Current one-step outputs refresh the targets as the generator changes. This objective follows TwinFlow's $\mathcal{L}_{\mathrm{adv}}^{\mathrm{TF}}$, with condition shifting selecting the fake trajectory. Gradients update the shared parameters through the shifted-condition prediction; generated samples and regression targets are detached when fitting the fake velocity field.

3.2 Rectifying the Generative Trajectory via Velocity Matching

Following DMD, evaluate the real and fake velocities at the same re-noised sample and time: $\widehat{\mathbf{v}}_{r,t'} := \boldsymbol{F}_\theta(\mathbf{x}_{\theta,t'}^{\mathrm{sg}}, t', \mathbf{c})$ and $\widehat{\mathbf{v}}_{\theta,t'} := \boldsymbol{F}_\theta(\mathbf{x}_{\theta,t'}^{\mathrm{sg}}, t', \mathbf{c}_{\mathrm{shift}})$. Under the OT score–velocity conversion (Appendix B.2), the real and fake velocity estimates are related to their induced scores by

$$\widehat{\mathbf{d}}_{t'} := \widehat{\mathbf{v}}_{\theta,t'} - \widehat{\mathbf{v}}_{r,t'} = -\frac{t'}{1-t'}\left(\widehat{\mathbf{s}}_{\theta,t'} - \widehat{\mathbf{s}}_{r,t'}\right). \tag{7}$$

If both velocities recover their marginal fields, this is proportional to $\mathbf{s}_{r,t'} - \mathbf{s}_{\theta,t'}$, the direction of the DMD correction in sample space. The hatted scores are network estimates, so this interpretation depends on the accuracy of both branches. After weighting and normalization, the correction $\widetilde{\mathbf{d}}_{t'}$ defines the following detached regression target for the one-step prediction

$$\mathcal{L}_{\mathrm{score}}(\theta) := \mathbb{E}\left[\left\|\boldsymbol{F}_\theta(\mathbf{z}, 1, \mathbf{c}) - \mathrm{sg}\left(\boldsymbol{F}_\theta(\mathbf{z}, 1, \mathbf{c}) - \widetilde{\mathbf{d}}_{t'}\right)\right\|_2^2\right]. \tag{8}$$

Since $\boldsymbol{G}_\theta = \mathbf{z} - \boldsymbol{F}_\theta(\mathbf{z}, 1, \mathbf{c})$, subtracting the correction from the velocity target adds it to the endpoint target. Gradients pass only through the prediction outside sg. The score identity motivates this update; it does not identify every weighted or normalized update with an exact KL gradient (Appendix B.3). The weighting and normalization determine how strongly the raw velocity difference contributes to the detached target for the one-step prediction.

3.3 Training Objective

With $\mathcal{L}_{\mathrm{base}} := \mathcal{L}_{\mathrm{RCGM}}$ (Sun & Lin, 2025), the generator objective is

$$\mathcal{L}_{\mathrm{APEX}}(\theta) = \mathcal{L}_{\mathrm{base}}(\theta) + \lambda_{\mathrm{score}}\mathcal{L}_{\mathrm{score}}(\theta). \tag{9}$$

The base objective supervises interval velocities, while $\lambda_{\mathrm{score}} \geq 0$ weights correction of the one-step outputs. Training generates samples under the original condition, re-noises them, and uses them to train the fake branch and form the detached correction. Equation (9) updates the generator, while $\mathcal{L}_{\mathrm{fake}}$ separately trains the shifted-condition prediction with shared parameters. Stop-gradient specifies the derivative paths within each loss; the branches do not have independent parameters. At inference, $\boldsymbol{G}_\theta(\mathbf{z}, \mathbf{c})$ uses only the original condition and requires one forward pass.

Table 1: System-level comparison of efficiency and quality. APEX uses SANA (0.6B/1.6B) and Qwen-Image (20B). Throughput (batch 10) and latency (batch 1) use a single A100 in BF16; baselines follow their cited protocols. Params gives total parameters; parentheses give LoRA rank r and trainable parameters (B). Total parameter counts are blue for full-parameter tuning and red for LoRA. † marks NFE-specific models. Within each NFE block, the best score for each metric is shown in **bold**.

Method	NFE	Throughput ↑ (samples/s)	Latency ↓ (s)	Params (B)	FID ↓	CLIP ↑	GenEval ↑
SDXL-LCM Luo et al. (2023)	2	2.89	0.40	0.9	18.11	27.51	0.44
PixArt-LCM Chen et al. (2024b)	2	3.52	0.31	0.6	10.33	27.24	0.42
PCM Wang et al. (2024)†	2	2.62	0.56	0.9	14.70	27.66	0.55
SDXL-DMD2 Yin et al. (2024a)†	2	2.89	0.40	0.9	7.61	**28.87**	0.58
FLUX-schnell (Labs, 2024)	2	0.92	1.15	12.0	7.75	28.25	0.71
Sana-Sprint (Chen et al., 2025b)	2	6.46	0.25	0.6	6.54	28.40	0.76
Sana-Sprint (Chen et al., 2025b)	2	5.68	0.24	1.6	6.50	28.45	0.77
Qwen-Image-Lightning (ModelTC, 2025)	2	3.15	0.48	20 (r=64,0.4)	6.76	28.37	0.85
RCGM (Sun & Lin, 2025)	2	3.15	0.48	20 (r=64,0.4)	6.80	28.63	0.82
TwinFlow (Cheng et al., 2025)	2	3.15	0.48	20 (r=64,0.4)	6.73	28.57	0.87
APEX	2	**6.50**	0.25	0.6	6.75	28.33	0.84
APEX	2	5.72	**0.23**	1.6	**6.42**	28.24	0.85
APEX	2	3.21	0.49	20 (r=32,0.2)	6.72	28.71	0.87
APEX	2	3.17	0.47	20 (r=64,0.4)	6.51	28.42	0.89
APEX	2	3.30	0.45	20	6.44	28.51	**0.90**
SDXL-LCM Luo et al. (2023)	1	3.36	0.32	0.9	50.51	24.45	0.28
PixArt-LCM Chen et al. (2024b)	1	4.26	0.25	0.6	73.35	23.99	0.41
PixArt-DMD Chen et al. (2024a)†	1	4.26	0.25	0.6	9.59	26.98	0.45
PCM Wang et al. (2024)†	1	3.16	0.40	0.9	30.11	26.47	0.42
SDXL-DMD2 Yin et al. (2024a)†	1	3.36	0.32	0.9	7.10	**28.93**	0.59
FLUX-schnell (Labs, 2024)	1	1.58	0.68	12.0	7.26	28.49	0.69
Sana-Sprint (Chen et al., 2025b)	1	7.22	0.21	0.6	7.04	28.04	0.72
Sana-Sprint (Chen et al., 2025b)	1	6.71	0.21	1.6	7.69	28.27	0.76
Qwen-Image-Lightning (ModelTC, 2025)	1	3.29	0.40	20 (r=64,0.4)	7.06	28.35	0.85
RCGM (Sun & Lin, 2025)	1	3.29	0.40	20 (r=64,0.4)	11.38	27.69	0.52
TwinFlow (Cheng et al., 2025)	1	3.29	0.40	20 (r=64,0.4)	7.32	28.29	0.86
APEX	1	**7.30**	**0.20**	0.6	6.99	28.36	0.84
APEX	1	6.84	**0.20**	1.6	**6.78**	28.12	0.84
APEX	1	3.29	0.39	20 (r=32,0.2)	7.22	28.62	0.88
APEX	1	3.27	0.39	20 (r=64,0.4)	7.14	28.45	**0.89**
APEX	1	3.50	0.34	20	6.87	28.66	**0.89**

4 EXPERIMENTS

We assess condition shifting through matched TwinFlow comparisons and alternative condition constructions, then examine practical training choices. System-level results establish quality and efficiency across model sizes and tuning modes; matched comparisons evaluate the methods with a common backbone, data, update budget, and evaluation protocol.

4.1 EXPERIMENTAL SETUP

Backbones and tuning. We use SANA 0.6B and 1.6B (Xie et al., 2024) with full-parameter tuning, and Qwen-Image 20B (Wu et al., 2025a) with full-parameter and LoRA tuning. The LoRA ranks $r = 32$ and $r = 64$ correspond to approximately 0.2B and 0.4B trainable parameters. We also compare APEX and TwinFlow on Z-Image 6B under a common training protocol.

Training data. We use ShareGPT-4o (Chen et al., 2025c) and BLIP-3o (Chen et al., 2025a), together with synthetic data generated by Qwen-Image 20B. The synthetic collection contains 600K samples from Flux-Reason-6M prompts (Fang et al., 2025) and 200K samples from poster prompts. Table 8 examines the effect of data and update budget.

Evaluation. GenEval (Ghosh et al., 2023) measures compositional alignment, complemented by FID and CLIP on MJHQ-30K (Li et al., 2024), DPG-Bench (Hu et al., 2024) for detailed prompt following, and WISE (Niu et al., 2025) for world-knowledge grounding. The Z-Image comparison

Table 2: Compositional alignment on GenEval. APEX uses 1 NFE; baseline sampling budgets follow the model labels or cited reports. Full denotes full-parameter tuning. Best per column in **bold**.

Model	Single object	Two objects	Counting	Colors	Position	Attribute binding	Overall ↑
PixArt-α (Chen et al., 2024c)	0.98	0.50	0.44	0.80	0.08	0.07	0.48
SD3 Medium (Esser et al., 2024)	0.98	0.74	0.63	0.67	0.34	0.36	0.62
FLUX.1 [Dev] (BlackForest, 2024)	0.98	0.81	0.74	0.79	0.22	0.45	0.66
SD3.5 Large (Esser et al., 2024)	0.98	0.89	0.73	0.83	0.34	0.47	0.71
Lumina-Image 2.0 (Qin et al., 2025)	-	0.87	0.67	-	-	0.62	0.73
Janus-Pro-7B (Chen et al., 2025d)	0.99	0.89	0.59	0.90	0.79	0.66	0.80
HiDream-I1-Full (Cai et al., 2025)	**1.00**	**0.98**	0.79	0.91	0.60	0.72	0.83
GPT Image 1 [High] (OpenAI, 2025)	0.99	0.92	0.85	0.92	0.75	0.61	0.84
Seedream 3.0 (Gao et al., 2025)	0.99	0.96	**0.91**	0.93	0.47	0.80	0.84
BAGEL (Deng et al., 2025)	0.98	0.95	0.84	**0.95**	0.78	0.77	0.88
Qwen-Image (Wu et al., 2025a)	0.99	0.92	0.89	0.88	0.76	0.77	0.87
Hyper-BAGEL (Lu et al., 2025)	0.97	0.86	0.75	0.90	0.67	0.62	0.80
Qwen-Image-Lightning (ModelTC, 2025)	0.99	0.89	0.85	0.87	0.75	0.76	0.85
TwinFlow (Cheng et al., 2025) (1 NFE)	**1.00**	0.91	0.84	0.90	0.75	0.74	0.86
APEX 0.6B	0.99	0.91	0.75	0.93	0.76	0.69	0.84
APEX 1.6B	0.99	0.91	0.75	0.93	0.76	0.68	0.84
APEX 20B (LoRA, $r = 32$)	0.99	0.95	0.85	0.90	0.79	0.78	0.88
APEX 20B (LoRA, $r = 64$)	0.99	0.94	0.88	0.90	0.85	0.78	**0.89**
APEX 20B (Full)	0.99	0.92	0.83	0.91	**0.86**	**0.81**	**0.89**

also uses OneIG (Chang et al., 2026) to assess multiple aspects of image generation and LongText-Bench (Geng et al., 2025b) for Chinese and English text rendering. All evaluations follow the protocols and aggregation rules of their benchmarks. Matched comparisons use the same evaluator and prompts for both methods. System-level APEX evaluations use 1024×1024 resolution; timing uses BF16 on one A100, with batch 10 for throughput and batch 1 for latency. Unless stated otherwise, APEX uses 1 NFE. Additional details are in Appendix A.5.

4.2 Efficiency and Performance Comparison

One-step quality and efficiency. At 1 NFE, APEX 0.6B achieves 0.84 GenEval with 7.30 samples/s throughput and 0.20-second latency (Table 1). Its GenEval score exceeds FLUX-Schnell 12B by 0.15, with $20\times$ fewer parameters. Against Sana-Sprint 0.6B, it improves GenEval from 0.72 to 0.84 at similar latency, while FID changes from 7.04 to 6.99. The 1.6B model retains 0.84 GenEval and lowers FID to 6.78. For Qwen-Image 20B, LoRA with $r = 32$ and $r = 64$ reaches 0.88 and 0.89 GenEval at 0.39-second latency. Full-parameter tuning also reaches 0.89, with a lower FID of 6.87; the tuning modes therefore differ even when their overall GenEval scores agree. The consistency-model baselines show larger quality changes as the sampling budget decreases. For SDXL-LCM, reducing 2 NFEs to 1 raises FID from 18.11 to 50.51 and lowers GenEval from 0.44 to 0.28; PCM shows the same direction of change. The corresponding one-step APEX models retain 0.84–0.89 GenEval, although the system-level comparison also includes differences in backbone and training protocol; the matched retrains provide a direct comparison with TwinFlow.

Effect of a second sampling step. The additional evaluation improves FID for every APEX configuration in Table 1. For the 1.6B model, FID falls from 6.78 to 6.42 and GenEval rises to 0.85, compared with 0.77 for Sana-Sprint 1.6B at similar latency. For 20B LoRA with $r = 64$, FID falls from 7.14 to 6.51 while GenEval remains 0.89 and latency rises from 0.39 to 0.47 seconds. Thus, a second step can improve distributional quality even when compositional accuracy changes little.

Compositional alignment. Table 2 separates the overall score into six categories. The 0.6B and 1.6B models perform strongly on single objects and colors, while counting and attribute binding leave more room for improvement. With 20B LoRA, increasing the rank from 32 to 64 improves counting from 0.85 to 0.88 and position from 0.79 to 0.85. Full tuning reaches 0.86 on position and 0.81 on attribute binding, the highest scores in those columns. Its counting score is 0.83, compared with 0.88 for $r = 64$ LoRA. The two models nevertheless share an Overall score of 0.89, with different strengths in counting, position, and attribute binding.

Table 3: Detailed prompt following on DPG-Bench. APEX uses 1 NFE; baseline sampling budgets follow the model labels or cited reports. Full denotes full-parameter tuning. Overall follows the benchmark's prompt-level aggregation. The best score in each column is shown in **bold**.

Model	Global	Entity	Attribute	Relation	Other	Overall ↑
PixArt-α (Chen et al., 2024c)	74.97	79.32	78.60	82.57	76.96	71.11
SDXL (Podell et al., 2023)	83.27	82.43	80.91	86.76	80.41	74.65
DALL-E 3 (OpenAI, 2023)	90.97	89.61	88.39	90.58	89.83	83.50
FLUX.1 [Dev] (BlackForest, 2024)	74.35	90.00	88.96	90.87	88.33	83.84
SD3-Medium (Esser et al., 2024)	87.90	91.01	88.83	80.70	88.68	84.08
HiDream-I1-Full (Cai et al., 2025)	76.44	90.22	89.48	93.74	91.83	85.89
Lumina-Image 2.0 (Qin et al., 2025)	-	91.97	90.20	**94.85**	-	87.20
Seedream 3.0 (Gao et al., 2025)	**94.31**	**92.65**	91.36	92.78	88.24	88.27
GPT Image 1 [High] (OpenAI, 2025)	88.89	88.94	89.84	92.63	90.96	85.15
Qwen-Image (Wu et al., 2025a)	91.32	91.56	92.02	94.31	**92.73**	**88.32**
Playground v3 (Liu et al., 2024)	91.94	85.71	90.90	90.00	92.72	87.04
TwinFlow (Cheng et al., 2025) (1 NFE)	92.34	92.12	**92.45**	92.86	92.63	86.52
APEX 0.6B	90.58	90.36	90.44	90.77	90.73	82.66
APEX 1.6B	90.77	90.56	90.63	90.98	90.94	83.22
APEX 20B (LoRA, $r = 32$)	93.12	90.95	91.38	90.65	91.73	86.17
APEX 20B (LoRA, $r = 64$)	92.46	91.14	90.71	91.30	91.98	85.77
APEX 20B (Full)	93.25	89.76	90.65	91.17	90.75	84.59

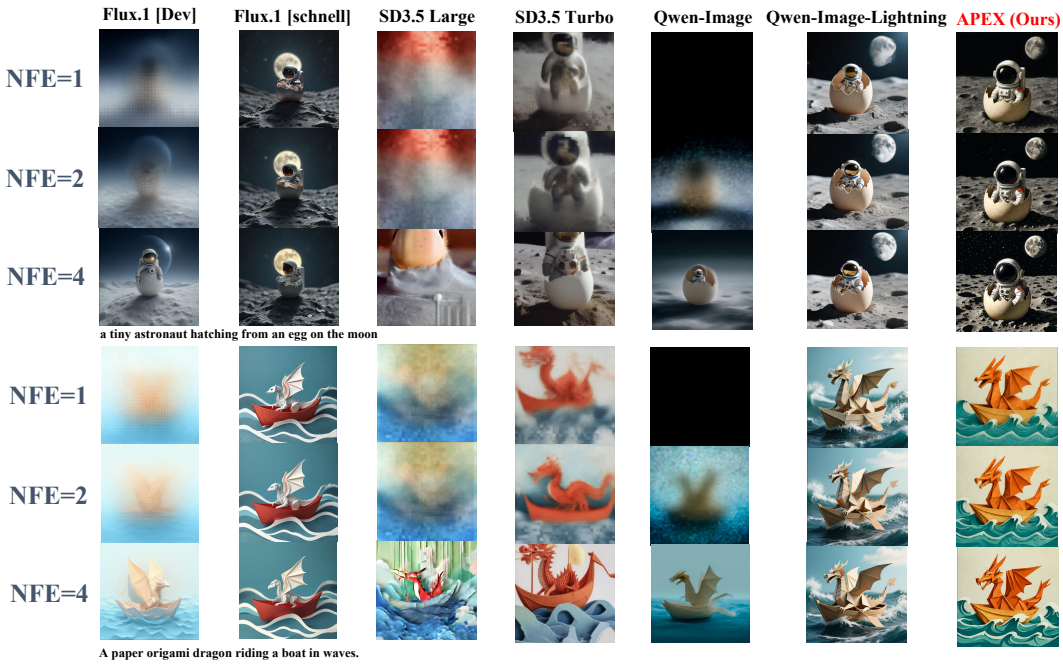

Figure 3: **Qualitative comparison across NFE budgets.** Samples from APEX and comparison models share prompts, seeds, and 1024×1024 resolution. Rows show 1, 2, and 4 NFEs; columns show models.

Detailed prompt following and world knowledge. The gains on GenEval do not imply uniform improvement across benchmarks. On DPG-Bench, $r = 32$ LoRA reaches 86.17, above 85.77 for $r = 64$ and 84.59 for full-parameter tuning (Table 3). The Qwen-Image teacher remains higher at 88.32. On WISE, one-step APEX matches TwinFlow's reported 0.54 Overall, with small differences across domains, while the teacher reaches 0.62 (Table 10). The broader baseline set places this result above SDXL (0.43), PixArt-α (0.47), and Qwen-Image-Lightning (0.51), while MetaQuery-XL reaches 0.55. The domain scores also differ: APEX is higher than MetaQuery-XL on Space, but lower on Cultural and Physics, despite their similar overall scores. Preserving compositional alignment therefore remains distinct from detailed prompt following and world-knowledge grounding.

Table 4: Matched APEX–TwinFlow comparison on Z-Image 6B at 2 NFE. Retrains share the training and evaluation protocol; reference results use the listed NFE budgets. CN/EN denote Chinese/English text rendering on LongText-Bench. Scores use a 0–100 scale. Best in the matched block in **bold**.

Method	NFE	OneIG ↑	GenEval ↑	DPG-Bench ↑	LongText	
					CN ↑	EN ↑
Matched retrains						
APEX (ours)	2	**52.60**	**77.23**	**88.64**	**90.38**	**88.76**
TwinFlow (Cheng et al., 2025) (retrain)	2	51.44	71.38	84.96	80.35	74.67
Reference results						
Z-Image-Turbo++ (Liu et al., 2026)	2	52.50	75.70	85.86	91.62	89.88
TwinFlow (Cheng et al., 2025)	2	51.38	72.41	85.98	78.65	71.99
Z-Image-Turbo	8	52.84	75.01	84.86	92.56	91.74

Figure 3 complements these scores with examples at 1, 2, and 4 NFEs using matched prompts, seeds, and resolution for each model in the qualitative comparison.

4.3 Matched Comparisons with TwinFlow

Z-Image 6B. To compare condition shifting with TwinFlow under a common protocol, we retrain both methods on Z-Image 6B with the same data, full-parameter tuning, global batch size 32, 12K updates, evaluator, and 2-NFE budget (Table 4). APEX scores higher on all five metrics: the gains are 1.16 points on OneIG, 5.85 on GenEval, 3.68 on DPG-Bench, 10.03 on LongText-CN, and 14.09 on LongText-EN. Text rendering shows the largest gains in both languages, while improvements on GenEval and DPG-Bench also indicate better compositional alignment and detailed prompt following. The separate reference block provides context from published results and official checkpoints with different training protocols. This comparison evaluates generation at 2 NFEs; the gains do not establish the same outcome for strict one-step generation on Z-Image.

Qwen-Image 20B. The additional matched retrains use Qwen-Image-Syn, full-parameter tuning, batch 32, 10K updates, and no prompt rewriting (Appendix A.3). At 1 NFE, APEX improves GenEval from 0.86 to 0.88 and WISE from 0.49 to 0.54, while DPG-Bench changes from 86.73 to 86.24. At 2 NFEs, the corresponding APEX scores are 0.89 GenEval, 86.45 DPG-Bench, and 0.58 WISE, compared with 0.86, 86.88, and 0.51 for TwinFlow. APEX improves compositional alignment and world knowledge at both budgets, with a small reduction in detailed prompt-following scores; these retrains are separate from the system-level checkpoints in Table 1 .

4.4 Ablations

We compare alternative conditions for the fake branch, vary the scale and bias of affine shifting, and examine the loss weights and training data. Results use 1 NFE and GenEval Overall unless stated otherwise. The endpoint-loss study uses the separate formulation in Appendix B.4 , rather than the velocity-regression objective in Eq. (9).

Alternative condition constructions. We test how the model identifies the fake trajectory using Qwen-Image 20B with full-parameter tuning on Qwen-Image-Syn at 1 NFE (Table 5). The affine construction uses $a = -0.5$, $b = 1$ and leads all three metrics. Relative to the discrete fake token, it gains 0.03 GenEval, 1.42 DPG-Bench points, and 0.01 WISE. Null conditioning removes text information, while Gaussian perturbation changes the condition on each call; both score lower. The results support affine condition shifting among the tested choices, whose retained information and perturbation magnitudes also differ between methods.

Table 5: Condition constructions for the fake branch. Qwen-Image 20B, full-parameter tuning, 1 NFE. Affine uses $a = -0.5$, $b = 1$; DPG denotes DPG-Bench. Best per column in **bold**.

Construction	GenEval	DPG	WISE
Scalar affine	**0.89**	**84.59**	**0.54**
Discrete fake token	0.86	83.17	0.53
Null conditioning	0.80	82.54	0.46
Gaussian perturbation	0.76	80.32	0.42

Condition shifting parameters. To examine scale and bias jointly, we vary a, b on SANA 0.6B using BLIP-3o, 3K updates, and 1 NFE (Table 6). Negation ($a = -1, b = 0$) gives 0.76; $a = -0.5, b = 1$ reaches 0.81. At $a = -1$, the tested biases give 0.73–0.76. For $a = 0.5$, increasing b to 10 raises the score to 0.73, but the same bias lowers the $a = -0.5$ result to 0.70. Thus, larger shifts do not consistently help. This grid and the Qwen-Image construction comparison use separate training settings and evaluate different model sizes.

Table 6: Condition shifting parameters. Rows vary scale a and columns bias b in $\mathbf{c}_{\text{shift}} = a\mathbf{c} + b\mathbf{1}$. GenEval at 1 NFE for APEX 0.6B on BLIP-3o after 3K updates. Best score in **bold**.

$a \backslash b$	**0.0**	**0.1**	**1.0**	**10.0**
-1.0	0.76	0.73	0.74	0.74
-0.5	0.75	0.79	**0.81**	0.70
0.5	0.29	0.37	0.30	0.73

Loss balance. This study uses the endpoint formulation in Appendix B.4, with BLIP-3o, 8K updates for 0.6B/1.6B, and 0.4K updates for 20B LoRA ($r = 32$). The fake-flow loss remains active while λ_r:λ_m varies (Table 7). Removing $\mathcal{L}_{\text{mix}}$ gives 0.32–0.42; equal weights reach 0.76–0.83, the highest scores in this comparison. The 0:1 setting removes only the explicit supervised term because $\mathcal{L}_{\text{mix}}$ also contains real-data supervision. These weights characterize the endpoint formulation and do not determine λ_{score} in Eq. (9).

Table 7: Endpoint loss-weight ablation. λ_r and λ_m weight the supervised and mixed endpoint losses in Appendix B.4. GenEval at 1 NFE; 20B uses $r = 32$ LoRA. The fake-flow loss remains active across all weight settings. Best per model in **bold**.

λ_r:λ_m	**0.6B**	**1.6B**	**20B**
1.0:0.0	0.32	0.35	0.42
0.0:1.0	0.63	0.66	0.69
1.0:0.5	0.72	0.71	0.81
1.0:1.0	**0.77**	**0.76**	**0.83**
1.0:2.0	0.74	0.75	0.82

Table 8: Effect of training data and update budget. GenEval Overall at 1 NFE on ShareGPT-4o and BLIP-3o. Qwen-Image 20B uses LoRA ($r = 32, 64$). Global batch size: 64. Best per row in **bold**.

Model	**ShareGPT-4o**			**BLIP-3o**		
	2K updates	8K updates	10K updates	2K updates	8K updates	10K updates
APEX 0.6B	0.37	0.67	0.73	0.71	0.77	**0.81**
APEX 1.6B	0.36	0.70	0.73	0.27	0.78	**0.83**
	0.4K updates	1K updates	2K updates	0.4K updates	1K updates	2K updates
APEX 20B (r=32)	0.19	0.33	0.62	0.83	**0.84**	0.83
APEX 20B (r=64)	0.21	0.35	0.61	0.73	**0.85**	0.84

Training data and update budget. We compare ShareGPT-4o and BLIP-3o at global batch 64 (Table 8). At 10K updates, BLIP-3o yields 0.81 and 0.83 for SANA 0.6B and 1.6B, versus 0.73 for both on ShareGPT-4o. For 20B LoRA, BLIP-3o reaches 0.84–0.85 at 1K updates, while ShareGPT-4o reaches 0.61–0.62 at 2K. On BLIP-3o, both LoRA ranks peak at 1K updates and change little at 2K, whereas the ShareGPT-4o scores continue to improve over the tested budgets.

5 Conclusion

We present **APEX**, a self-adversarial framework that learns real and fake trajectories in a shared network through condition shifting. Their velocity discrepancy supplies an internal correction that complements trajectory-based training while retaining one-pass inference. Across 0.6B–20B text-to-image models, APEX achieves strong GenEval results with sub-second latency and supports both full-parameter and LoRA tuning. Under matched training, APEX improves all five reported metrics on Z-Image at 2 NFEs, with the largest gains in Chinese and English text rendering. On Qwen-Image, it improves GenEval and WISE at both budgets, with slightly lower DPG-Bench scores. The condition ablations favor affine shifting among the tested constructions. These results support learning the fake trajectory through the condition input while preserving the original generation task. The accuracy of both velocity estimates remains important for the correction; further evaluation across training seeds and on video generation would help establish the scope of this approach.

AI USE STATEMENT

We used generative AI tools to assist manuscript writing and revision. The training data include synthetic images generated by Qwen-Image 20B, as described in Section 4.1. We take responsibility for the final manuscript and its scientific claims.

ETHICS STATEMENT

APEX aims to reduce the computational cost of text-to-image generation. Faster generation can also lower the cost of producing deceptive or harmful visual content. Applications should retain content-safety safeguards, disclose synthetic media when its origin affects interpretation, and respect the licenses and terms of the underlying models and datasets. Deployment requires assessment of biases and harmful content associated with the base models.

REFERENCES

BlackForest. Flux. https://github.com/black-forest-labs/flux, 2024.

Qi Cai, Jingwen Chen, Yang Chen, Yehao Li, Fuchen Long, Yingwei Pan, Zhaofan Qiu, Yiheng Zhang, Fengbin Gao, Peihan Xu, et al. Hidream-i1: A high-efficient image generative foundation model with sparse diffusion transformer. *arXiv preprint arXiv:2505.22705*, 2025.

Jingjing Chang, Yixiao Fang, Peng Xing, Shuhan Wu, Wei Cheng, Rui Wang, Xianfang Zeng, Gang Yu, and Hai-Bao Chen. Oneig-bench: Omni-dimensional nuanced evaluation for image generation. *Advances in Neural Information Processing Systems*, 38, 2026.

Jiuhai Chen, Zhiyang Xu, Xichen Pan, Yushi Hu, Can Qin, Tom Goldstein, Lifu Huang, Tianyi Zhou, Saining Xie, Silvio Savarese, et al. Blip3-o: A family of fully open unified multimodal models-architecture, training and dataset. *arXiv preprint arXiv:2505.09568*, 2025a.

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

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

Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Zhongdao Wang, James T Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *ICLR*, 2024c.

Junsong Chen, Shuchen Xue, Yuyang Zhao, Jincheng Yu, Sayak Paul, Junyu Chen, Han Cai, Enze Xie, and Song Han. Sana-sprint: One-step diffusion with continuous-time consistency distillation. *arXiv preprint arXiv:2503.09641*, 2025b.

Junying Chen, Zhenyang Cai, Pengcheng Chen, Shunian Chen, Ke Ji, Xidong Wang, Yunjin Yang, and Benyou Wang. Sharegpt-4o-image: Aligning multimodal models with gpt-4o-level image generation. *arXiv preprint arXiv:2506.18095*, 2025c.

Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*, 2025d.

Zhenglin Cheng, Peng Sun, Jianguo Li, and Tao Lin. Twinflow: Realizing one-step generation on large models with self-adversarial flows. *arXiv preprint arXiv:2512.05150*, 2025.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, Guang Shi, and Haoqi Fan. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

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

Rongyao Fang, Aldrich Yu, Chengqi Duan, Linjiang Huang, Shuai Bai, Yuxuan Cai, Kun Wang, Si Liu, Xihui Liu, and Hongsheng Li. Flux-reason-6m & prism-bench: A million-scale text-to-image reasoning dataset and comprehensive benchmark. *arXiv preprint arXiv:2509.09680*, 2025.

Yu Gao, Lixue Gong, Qiushan Guo, Xiaoxia Hou, Zhichao Lai, Fanshi Li, Liang Li, Xiaochen Lian, Chao Liao, Liyang Liu, et al. Seedream 3.0 technical report. *arXiv preprint arXiv:2504.11346*, 2025.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025a.

Zigang Geng, Yibing Wang, Yeyao Ma, Chen Li, Yongming Rao, Shuyang Gu, Zhao Zhong, Qinglin Lu, Han Hu, Xiaosong Zhang, et al. X-omni: Reinforcement learning makes discrete autoregressive image generative models great again. *arXiv preprint arXiv:2507.22058*, 2025b.

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. 2014.

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

Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. Ella: Equip diffusion models with llm for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 2024.

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

Dongjun Kim, Yeongmin Kim, Se Jung Kwon, Wanmo Kang, and Il-Chul Moon. Refining generative process with discriminator guidance in score-based diffusion models. In *International Conference on Machine Learning*, pp. 16567–16598. PMLR, 2023.

Youngjoong Kim, Deokyeong Lee, and Jaesik Park. Distribution matching distillation without fake score network. *arXiv preprint arXiv:2605.19256*, 2026.

Black Forest Labs. Flux, 2024. URL https://blackforestlabs.ai/.

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

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

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

Deyuan Liu, Peng Sun, Xufeng Li, and Tao Lin. Efficient generative model training via embedded representation warmup. *arXiv preprint arXiv:2504.10188*, 2025.

Dongyang Liu, Ruoyi Du, David Liu, Dengyang Jiang, Liangchen Li, Qilong Wu, Zhen Li, Steven C. H. Hoi, Hongsheng Li, and Peng Gao. High-fidelity two-step image generation via teacher-aligned end-to-end distillation. *arXiv preprint arXiv:2606.12575*, 2026.

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

Yanzuo Lu, Xin Xia, Manlin Zhang, Huafeng Kuang, Jianbin Zheng, Yuxi Ren, and Xuefeng Xiao. Hyper-bagel: A unified acceleration framework for multimodal understanding and generation, 2025.

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

ModelTC. Qwen-image-lightning. GitHub-ModelTC/Qwen-Image-Lightning: Qwen-Image-Lightning:SpeedupQwen-Imagemodelwithdistilla, 2025.

Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

Yuwei Niu, Munan Ning, Mengren Zheng, Weiyang Jin, Bin Lin, Peng Jin, Jiaqi Liao, Chaoran Feng, Kunpeng Ning, Bin Zhu, et al. Wise: A world knowledge-informed semantic evaluation for text-to-image generation. *arXiv preprint arXiv:2503.07265*, 2025.

OpenAI. Dalle-3, 2023. URL https://openai.com/dall-e-3.

OpenAI. Gpt-image-1, 2025. URL https://openai.com/index/introducing-4o-image-generation/.

Xichen Pan, Satya Narayan Shukla, Aashu Singh, Zhuokai Zhao, Shlok Kumar Mishra, Jialiang Wang, Zhiyang Xu, Jiuhai Chen, Kunpeng Li, Felix Juefei-Xu, et al. Transfer between modalities with metaqueries. *arXiv preprint arXiv:2504.06256*, 2025.

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

Qi Qin, Le Zhuo, Yi Xin, Ruoyi Du, Zhen Li, Bin Fu, Yiting Lu, Jiakang Yuan, Xinyue Li, Dongyang Liu, et al. Lumina-image 2.0: A unified and efficient image generative framework. *arXiv preprint arXiv:2503.21758*, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. 2022.

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

Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation. *arXiv preprint arXiv:2403.12015*, 2024.

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

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

Peng Sun and Tao Lin. Any-step generation via n-th order recursive consistent velocity field estimation, 2025. URL https://github.com/LINs-lab/RCGM. GitHub repository.

Peng Sun, Yi Jiang, and Tao Lin. Unified continuous generative models. *arXiv preprint arXiv:2505.07447*, 2025.

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

Zidong Wang, Yiyuan Zhang, Xiaoyu Yue, Xiangyu Yue, Yangguang Li, Wanli Ouyang, and Lei Bai. Transition models: Rethinking the generative learning objective. *arXiv preprint arXiv:2509.04394*, 2025.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025a.

Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, et al. Janus: Decoupling visual encoding for unified multimodal understanding and generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 12966–12977, 2025b.

Yecheng Wu, Zhuoyang Zhang, Junyu Chen, Haotian Tang, Dacheng Li, Yunhao Fang, Ligeng Zhu, Enze Xie, Hongxu Yin, Li Yi, et al. Vila-u: a unified foundation model integrating visual understanding and generation. *arXiv preprint arXiv:2409.04429*, 2024.

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

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

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6613–6623, 2024b.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

Kaiwen Zheng, Yongxin Chen, Huayu Chen, Guande He, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Direct discriminative optimization: Your likelihood-based visual generative model is secretly a gan discriminator. *arXiv preprint arXiv:2503.01103*, 2025a.

Kaiwen Zheng, Yuji Wang, Qianli Ma, Huayu Chen, Jintao Zhang, Yogesh Balaji, Jianfei Chen, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Large scale diffusion distillation via score-regularized continuous-time consistency. *arXiv preprint arXiv:2510.08431*, 2025b.

Mingyuan Zhou, Yi Gu, and Zhendong Wang. Few-step diffusion via score identity distillation. *arXiv preprint arXiv:2505.12674*, 2025.

CONTENTS

1 Introduction **2**

2 Preliminaries **3**

3 APEX **3**
3.1 Condition Shifting for Self-Adversarial Training 3
3.2 Rectifying the Generative Trajectory via Velocity Matching 4
3.3 Training Objective 4

4 Experiments **5**
4.1 Experimental Setup 5
4.2 Efficiency and Performance Comparison 6
4.3 Matched Comparisons with TwinFlow 8
4.4 Ablations 8

5 Conclusion **9**

A Related Work **14**
A.1 Consistency and Trajectory-Based Generation 14
A.2 From External Discriminators to Internal Correction 15
A.3 Matched APEX–TwinFlow Comparison on Qwen-Image 15
A.4 World-Knowledge Grounding on WISE 15
A.5 Additional Experimental Details 15
A.6 Scalable Training 16

B Theoretical Analysis and Proofs **16**
B.1 Setup 17
B.2 Score–Velocity Duality under OT Path 17
B.3 Frozen-Target Velocity Correction 19
B.4 Endpoint Formulation for the Loss Ablations 20
B.5 Endpoint–Velocity Equivalence 20
B.6 Gradient Equivalence of Alternative Loss 21
B.7 Score-Difference Form of the Endpoint Surrogate Gradient 23
B.8 GAN Dynamics and Score-Difference Gradients 24

C Visualizations Part I **25**

D Visualizations Part II **27**

E Visualizations Part III **32**

F Limitations **34**

A RELATED WORK

A.1 CONSISTENCY AND TRAJECTORY-BASED GENERATION

Diffusion (Ho et al., 2020; Song et al., 2020; Karras et al., 2022) and flow matching (Lipman et al., 2022; Liu et al., 2025) generate samples by integrating a learned velocity field from noise to data. Using fewer integration steps can increase discretization error and reduce sample quality (Karras et al., 2022). Consistency and trajectory-based methods train the model to predict over a time interval, allowing generation with fewer network evaluations. Consistency Models (CMs) (Song et al., 2023; Lu & Song, 2024) learn to map points on the same trajectory to a common endpoint. MeanFlow (Geng et al., 2025a) predicts the average velocity over an interval, while Transition Models (TiM) (Wang et al., 2025) learn the transition between its endpoints. UCGM (Sun et al., 2025) unifies different paradigms by interpolating between their respective training objectives with a hyperparameter. APEX complements the RCGM base objective with internal distribution matching. The fake-flow loss $\mathcal{L}_{\text{fake}}$ (Eq. (6)) learns the velocity field of generated samples, supplying the

Table 9: Matched comparison on Qwen-Image 20B. Both methods use full-parameter tuning on Qwen-Image-Syn for 10K updates, with batch size 32 and no prompt rewriting. Retrains are separate from the checkpoints in Table 1. Best score for each metric and NFE in **bold**.

Method	Updates	NFE	GenEval ↑	DPG-Bench ↑	WISE ↑
TwinFlow (Cheng et al., 2025) (retrain)	10K	1	0.86	**86.73**	0.49
APEX (retrain)	10K	1	**0.88**	86.24	**0.54**
TwinFlow (Cheng et al., 2025) (retrain)	10K	2	0.86	**86.88**	0.51
APEX (retrain)	10K	2	**0.89**	86.45	**0.58**

fake-trajectory estimate used for correction. The resulting velocity difference supplies the frozen regression target for $\mathcal{L}_{\text{score}}$, while $\mathcal{L}_{\text{base}}$ provides the RCGM trajectory objective.

A.2 From External Discriminators to Internal Correction

Achieving high one-step fidelity benefits from direct anchoring of the endpoint prediction to the data manifold, a property that relative consistency constraints alone may not guarantee. External correction signals can come from auxiliary score models, as in distribution matching (Yin et al., 2024b;a), or from discriminators used in adversarial refiners (Kim et al., 2023; Sauer et al., 2024; Zheng et al., 2025a). Additional networks increase memory use and update cost. Some early distillation methods also use precomputed teacher–student pairs for regularization, while DMD2 removes that paired-data requirement (Yin et al., 2024a). Other methods obtain correction signals from the generative model itself. Direct Discriminative Optimization (DDO) (Zheng et al., 2025a) expresses the GAN discriminator through the likelihood ratio between a target model and a fixed reference model.

TwinFlow (Cheng et al., 2025) learns real and fake trajectories within a shared model and uses their velocity discrepancy to construct an internal correction signal.

APEX follows this line and separates the branches through condition shifting. The shifted condition identifies the fake trajectory for the original prompt, and the difference between the fake and real velocity estimates supplies the correction in Section 3.2 .

Few-step score identity distillation (SiD) (Zhou et al., 2025) also studies how conditioning affects distribution matching. Its Zero-CFG strategy removes text conditioning from the fake-score network, while Anti-CFG applies negative classifier-free guidance by combining conditional and unconditional predictions. These strategies act on the fake-score estimate. APEX trains the shared network on generated samples under a shifted text condition to estimate the fake velocity.

Fake-Score-network-Free DMD (FSF-DMD) (Kim et al., 2026) replaces the auxiliary fake-score network with an endpoint pseudo-velocity obtained from a flow-map generator. It shares the aim of reducing auxiliary-network overhead. The estimator differs: FSF-DMD uses a generator-induced surrogate, whereas APEX explicitly learns a fake velocity branch through flow matching on current generated samples under a shifted text condition.

A.3 Matched APEX–TwinFlow Comparison on Qwen-Image

Table 4 in the main paper matches the two constructions on Z-Image. We additionally compare the two methods on Qwen-Image-20B under matched training. We retrain both methods under the same protocol: Qwen-Image 20B, Qwen-Image-Syn, full-parameter tuning, global batch 32, 10K updates, the same evaluator, and no LLM prompt rewriting. Published TwinFlow results use different training data and are listed separately in the main tables.

Table 9 reports the complete results for both methods and sampling budgets.

Under this protocol APEX is higher on GenEval and WISE at both NFE, and TwinFlow is higher on DPG-Bench at both NFE (by 0.49 and 0.43). The lower DPG-Bench scores differ from the gains on Z-Image, where APEX improves all five metrics under matched training.

A.4 World-Knowledge Grounding on WISE

Table 10 reports the per-domain scores discussed in Section 4.2 .

A.5 Additional Experimental Details

The system-level APEX entries in Table 1 use 1024×1024 evaluation. Throughput and latency are measured in BF16 on a single A100 at batch sizes 10 and 1, respectively. Published baselines

Table 10: World-knowledge grounding on WISE. Per-domain and overall scores. APEX uses Qwen-Image 20B with full-parameter tuning (Full) at 1 NFE; baselines use their reported sampling budgets. The highest score in each column is shown in **bold**.

Model	Cultural	Time	Space	Biology	Physics	Chemistry	Overall ↑
SD v1.5 (Rombach et al., 2022)	0.34	0.35	0.32	0.28	0.29	0.21	0.32
SDXL (Podell et al., 2023)	0.43	0.48	0.47	0.44	0.45	0.27	0.43
PixArt-α (Chen et al., 2024c)	0.45	0.50	0.48	0.49	0.56	0.34	0.47
Playground-v2.5 (Li et al., 2024)	0.49	0.58	0.55	0.43	0.48	0.33	0.49
FLUX.1 [Dev] (BlackForest, 2024)	0.48	0.58	0.62	0.42	0.51	0.35	0.50
Janus (Wu et al., 2025b)	0.16	0.26	0.35	0.28	0.30	0.14	0.23
Janus-Pro-7B (Chen et al., 2025d)	0.30	0.37	0.49	0.36	0.42	0.26	0.35
VILA-U (Wu et al., 2024)	0.51	0.51	0.51	0.49	0.51	0.49	0.50
MetaQuery-XL (Pan et al., 2025)	0.56	0.55	0.62	0.49	0.63	0.41	0.55
BAGEL (Deng et al., 2025)	0.44	0.55	0.68	0.44	0.60	0.39	0.52
GPT-4o	**0.81**	**0.71**	**0.89**	**0.83**	**0.79**	**0.74**	**0.80**
Qwen-Image (Wu et al., 2025a)	-	-	-	-	-	-	0.62
Qwen-Image-Lightning (ModelTC, 2025)	-	-	-	-	-	-	0.51
TwinFlow (Cheng et al., 2025)	0.52	0.51	0.67	0.48	0.61	0.40	0.54
APEX 20B (Full)	0.53	0.54	0.66	0.48	0.61	0.41	0.54

retain their cited evaluation protocols; the matched retrains in Table 4 and Table 9 are separate comparisons with matched data, update budgets, and evaluation protocols.

A representative Qwen-Image 20B LoRA run used 2K updates and global batch size 64, taking approximately six hours. This timing is reported separately from the checkpoints in Table 1, since the hardware and parallelism records were not paired with those evaluations.

For the loss-weight ablation, the training data are BLIP-3o, with 8K updates for the 0.6B and 1.6B models and 0.4K updates for the 20B LoRA model. The condition-parameter grid uses the 0.6B model after 3K updates. The condition-construction comparison uses Qwen-Image 20B with full-parameter tuning on Qwen-Image-Syn. The loss-weight ablation uses the endpoint definitions in Appendix B.4; its coefficients are separate from the velocity-regression weight in Eq. (9).

The affine variants use $\mathbf{A} = a\mathbf{I}$ and $\mathbf{b} = b\mathbf{1}$, giving $\mathbf{c}_{\text{shift}} = a\mathbf{c} + b\mathbf{1}$. For the Gaussian construction, $\mathbf{c}_{\text{shift}} = \mathbf{c} + \boldsymbol{\eta}$, where $\boldsymbol{\eta}$ has the same shape as $\mathbf{c}$ and independent standard normal entries. It is resampled per example and forward pass before text normalization and input projection. The symbol $\boldsymbol{\eta}$ denotes condition-space noise, distinct from the image-space noise $\boldsymbol{\epsilon}'$ used to re-noise generated samples before estimating the fake velocity field.

A.6 Scalable Training

Computing time derivatives is a practical challenge when training continuous-time consistency models at large scale. Methods such as MeanFlow (Geng et al., 2025a) use Jacobian-vector products (JVP) to compute these derivatives. Efficient JVP computation with FlashAttention (Dao, 2023) and FSDP (Zhao et al., 2023) requires support in the attention kernels and distributed implementation. rCM (Zheng et al., 2025b) provides this support through a dedicated FlashAttention-2 JVP kernel and a compatible layer implementation. In APEX, the real and fake velocities are obtained through ordinary forward passes of the shared network. Finite-difference estimators provide a scalable alternative (Lu & Song, 2024; Wang et al., 2025), using forward evaluations to approximate the required derivatives. APEX uses the RCGM base objective alongside the internal velocity-matching correction. The correction requires no JVP computation and is compatible with both full-parameter and LoRA tuning, as evaluated on SANA and Qwen-Image in the main paper.

B Theoretical Analysis and Proofs

We derive the OT score–velocity identity and the gradient of the frozen-target correction in Section 3. The endpoint formulation in Appendix B.4 defines the objectives used in the loss ablations. These differ from Eq. (8), which corrects predictions from pure-noise inputs.

B.1 SETUP

We use bold lowercase letters for vectors like $\mathbf{x}, \mathbf{z}, \mathbf{v}$ and bold uppercase letters for matrices and operators like $\boldsymbol{F}$. The identity matrix is denoted by $\mathbf{I}$, and $\mathbf{0}$ represents the zero vector. Let $p_r(\mathbf{x})$ denote the data distribution over $\mathbf{x} \in \mathbb{R}^d$, and let $p_z(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ be the standard Gaussian prior over $\mathbf{z} \in \mathbb{R}^d$. For conditional generation, $\mathbf{c}$ denotes the text embedding, and $p_r(\mathbf{x} \mid \mathbf{c})$ is the conditional data distribution. We omit $\mathbf{c}$ when it is fixed. Throughout this appendix, we work with the Optimal Transport (OT) interpolation path defined by:

$$\mathbf{x}_t = \alpha(t)\mathbf{z} + \gamma(t)\mathbf{x}, \quad t \in [0, 1], \tag{10}$$

where $\alpha(t) = t$ and $\gamma(t) = 1 - t$. This satisfies the boundary conditions: $\mathbf{x}_0 = \mathbf{x}$ (pure data) and $\mathbf{x}_1 = \mathbf{z}$ (pure noise). All score identities and time weights below are used for $t \in (0, 1)$. Given a time-dependent random variable $\mathbf{x}_t$ following Eq. (10), we define the conditional mean velocity. The per-sample FM target is the constant velocity along this linear interpolation:

$$\mathbf{u}_r(\mathbf{x}, \mathbf{z}) := \mathbf{z} - \mathbf{x}. \tag{11}$$

The population real-data velocity is the conditional mean of this target:

$$\mathbf{v}_r^*(\mathbf{x}_t, t) := \mathbb{E}[\mathbf{u}_r(\mathbf{x}, \mathbf{z}) \mid \mathbf{x}_t, t]. \tag{12}$$

The score function is $\mathbf{s}_{r,t}(\mathbf{x}_t) := \nabla_{\mathbf{x}_t} \log p_{r,t}(\mathbf{x}_t)$, where $p_{r,t}(\mathbf{x}_t)$ is the marginal density of $\mathbf{x}_t$ at time t. A star marks a population conditional mean, distinct from a network estimate. We parameterize a velocity field estimator by a neural network $\boldsymbol{F}_\theta : \mathbb{R}^d \times [0, 1] \times \mathcal{C} \to \mathbb{R}^d$, where θ denotes the model parameters and $\mathcal{C}$ is the conditioning space. We use the shorthand $\boldsymbol{F}_\theta(\mathbf{x}_t, t, \mathbf{c}) \equiv \boldsymbol{F}_\theta$ when the arguments are clear from context. The operator $\mathrm{sg}(\cdot)$ denotes stop-gradient. For $\theta \in \mathbb{R}^P$ and a vector output $\mathbf{h} \in \mathbb{R}^d$, $J_\theta \mathbf{h} := \partial \mathbf{h} / \partial \theta \in \mathbb{R}^{d \times P}$, so parameter gradients use $J_\theta \mathbf{h}$ transposed. The branch indices r and θ identify original- and shifted-condition velocity estimates throughout. A hat denotes a network estimate; an unadorned score denotes a true marginal score.

Interval and instantaneous calls. To make the suppressed RCGM destination time explicit, write the shared predictor as $\mathcal{F}_\theta(\mathbf{y}, t, s, \mathbf{c})$, where $s \le t$. Its interval update is

$$\widehat{\mathbf{x}}_s = \mathbf{y} - (t - s)\mathcal{F}_\theta(\mathbf{y}, t, s, \mathbf{c}).$$

RCGM supervises this interval-averaged velocity; the diagonal call $s = t$ supplies the instantaneous velocity used in flow matching and score conversion. In the main text, $\boldsymbol{F}_\theta(\mathbf{y}, t, \mathbf{c})$ suppresses this diagonal argument in velocity evaluations, whereas the one-step prediction $\boldsymbol{F}_\theta(\mathbf{z}, 1, \mathbf{c})$ abbreviates $\mathcal{F}_\theta(\mathbf{z}, 1, 0, \mathbf{c})$. Thus $\boldsymbol{G}_\theta(\mathbf{z}, \mathbf{c}) = \mathbf{z} - \mathcal{F}_\theta(\mathbf{z}, 1, 0, \mathbf{c})$ uses the interval $[0, 1]$. These calls share parameters but need not return equal velocities. The endpoint-loss identities below use the velocity evaluation specified in each loss and do not assume equality with an interval average.

We define the endpoint predictor that maps a velocity estimate to its implied clean sample:

$$\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}, \mathbf{x}_t, t) := \mathbf{x}_t - t \cdot \boldsymbol{F}. \tag{13}$$

This is motivated by the OT path: if $\mathbf{x}_t = t\mathbf{z} + (1 - t)\mathbf{x}$ and $\boldsymbol{F} \approx \mathbf{z} - \mathbf{x}$, then $\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}, \mathbf{x}_t, t) \approx \mathbf{x}$.

B.2 SCORE–VELOCITY DUALITY UNDER OT PATH

We establish a useful relationship between the score function and the optimal velocity field under the OT path for both real and generated endpoint distributions.

Proposition 1 (Score–Velocity Duality) . *Let $\mathbf{x}_t = t\mathbf{z} + (1 - t)\mathbf{x}$ for $t \in (0, 1)$, where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is independent of $\mathbf{x} \sim p_r$. Denote by $p_{r,t}(\mathbf{x}_t)$ the marginal density of $\mathbf{x}_t$, and define the OT-induced conditional mean (least-squares optimal) velocity field*

$$\mathbf{v}_r^*(\mathbf{x}_t, t) := \mathbb{E}[\mathbf{u}_r(\mathbf{x}, \mathbf{z}) \mid \mathbf{x}_t, t]. \tag{14}$$

Then the score function $\mathbf{s}_{r,t}(\mathbf{x}_t) := \nabla_{\mathbf{x}_t} \log p_{r,t}(\mathbf{x}_t)$ satisfies

$$\mathbf{s}_{r,t}(\mathbf{x}_t) = -\frac{\mathbf{x}_t + (1 - t)\mathbf{v}_r^*(\mathbf{x}_t, t)}{t}. \tag{15}$$

Proof. **Step 1: Rewrite as an additive Gaussian observation model.** Define $\mathbf{x}' := (1-t)\mathbf{x}$. Then the OT path becomes an additive Gaussian observation of the scaled endpoint:

$$\mathbf{x}_t = \mathbf{x}' + t\mathbf{z}, \qquad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

Conditioned on $\mathbf{x}'$, the likelihood is $\mathbf{x}_t \mid \mathbf{x}' \sim \mathcal{N}(\mathbf{x}', t^2\mathbf{I})$, since $\mathbf{x}_t - \mathbf{x}' = t\mathbf{z}$ and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ implies $t\mathbf{z} \sim \mathcal{N}(\mathbf{0}, t^2\mathbf{I})$, with covariance determined by the sampled time.

Step 2: Apply Tweedie's formula to recover the posterior mean. For an additive Gaussian model $\mathbf{x}_t = \mathbf{x}' + t\mathbf{z}$ where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, Tweedie's formula states that the posterior mean can be recovered from the score function of the marginal distribution at that noising time:

$$\mathbb{E}[\mathbf{x}' \mid \mathbf{x}_t] = \mathbf{x}_t + t^2 \nabla_{\mathbf{x}_t} \log p_{r,t}(\mathbf{x}_t) = \mathbf{x}_t + t^2\, \mathbf{s}_{r,t}(\mathbf{x}_t). \tag{16}$$

Applying Tweedie's formula: For a Gaussian perturbation model $\mathbf{y} = \mathbf{x}' + \sigma\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the posterior mean of the clean variable is related to the marginal score by

$$\mathbb{E}[\mathbf{x}' \mid \mathbf{y}] = \mathbf{y} + \sigma^2 \nabla_{\mathbf{y}} \log p(\mathbf{y}).$$

In our case, $\mathbf{y} = \mathbf{x}_t$, $\mathbf{x}' = (1-t)\mathbf{x}$, and $\sigma = t$, so Eq. (16) follows directly.

Since $\mathbf{x}' = (1-t)\mathbf{x}$, we can recover the conditional expectation of $\mathbf{x}$:

$$\begin{aligned}
\mathbb{E}[\mathbf{x} \mid \mathbf{x}_t] &= \frac{1}{1-t}\mathbb{E}[\mathbf{x}' \mid \mathbf{x}_t] \\
&= \frac{1}{1-t}\left[\mathbf{x}_t + t^2\, \mathbf{s}_{r,t}(\mathbf{x}_t)\right] \\
&= \frac{\mathbf{x}_t + t^2\, \mathbf{s}_{r,t}(\mathbf{x}_t)}{1-t}.
\end{aligned} \tag{17}$$

Step 3: Express the conditional mean of $\mathbf{z}$. From the OT path $\mathbf{x}_t = t\mathbf{z} + (1-t)\mathbf{x}$, we can solve for $\mathbf{z}$ in terms of the noised sample and its clean endpoint:

$$\mathbf{z} = \frac{\mathbf{x}_t - (1-t)\mathbf{x}}{t}.$$

Taking conditional expectations on both sides given $\mathbf{x}_t$:

$$\begin{aligned}
\mathbb{E}[\mathbf{z} \mid \mathbf{x}_t] &= \mathbb{E}\left[\frac{\mathbf{x}_t - (1-t)\mathbf{x}}{t} \,\middle|\, \mathbf{x}_t\right] \\
&= \frac{1}{t}\Big[\mathbf{x}_t - (1-t)\mathbb{E}[\mathbf{x} \mid \mathbf{x}_t]\Big] \\
&= \frac{1}{t}\left[\mathbf{x}_t - (1-t)\cdot\frac{\mathbf{x}_t + t^2\, \mathbf{s}_{r,t}(\mathbf{x}_t)}{1-t}\right] \quad \text{(substituting Eq. (17))} \\
&= \frac{1}{t}\Big[\mathbf{x}_t - \big(\mathbf{x}_t + t^2\, \mathbf{s}_{r,t}(\mathbf{x}_t)\big)\Big] \quad \text{(simplifying the fraction)} \\
&= \frac{1}{t}\left[-t^2\, \mathbf{s}_{r,t}(\mathbf{x}_t)\right] \\
&= -t\, \mathbf{s}_{r,t}(\mathbf{x}_t).
\end{aligned} \tag{18}$$

Step 4: Form the optimal velocity and rearrange. By definition, the (least-squares) optimal velocity field along the OT path is the conditional expectation of the target velocity $\mathbf{z} - \mathbf{x}$:

$$\mathbf{v}_r^*(\mathbf{x}_t, t) := \mathbb{E}[\mathbf{u}_r(\mathbf{x}, \mathbf{z}) \mid \mathbf{x}_t, t] = \mathbb{E}[\mathbf{z} \mid \mathbf{x}_t] - \mathbb{E}[\mathbf{x} \mid \mathbf{x}_t].$$

Substituting Eq. (17) and Eq. (18) into the conditional-mean velocity definition gives

$$\begin{aligned}
\mathbf{v}_r^*(\mathbf{x}_t, t) &= -t\, \mathbf{s}_{r,t}(\mathbf{x}_t) - \frac{\mathbf{x}_t + t^2\, \mathbf{s}_{r,t}(\mathbf{x}_t)}{1-t} \\
&= \frac{-t(1-t)\, \mathbf{s}_{r,t}(\mathbf{x}_t) - (\mathbf{x}_t + t^2\, \mathbf{s}_{r,t}(\mathbf{x}_t))}{1-t} \quad \text{(common denominator)} \\
&= \frac{-t\, \mathbf{s}_{r,t}(\mathbf{x}_t) + t^2\, \mathbf{s}_{r,t}(\mathbf{x}_t) - \mathbf{x}_t - t^2\, \mathbf{s}_{r,t}(\mathbf{x}_t)}{1-t} \\
&= \frac{-t\, \mathbf{s}_{r,t}(\mathbf{x}_t) - \mathbf{x}_t}{1-t} \\
&= -\frac{\mathbf{x}_t + t\, \mathbf{s}_{r,t}(\mathbf{x}_t)}{1-t}.
\end{aligned} \tag{19}$$

Step 5: Rearrange to obtain the score-velocity duality. Multiplying both sides of Eq. (19) by $(1-t)$ gives an equivalent relation between the velocity and score:

$$(1-t)\mathbf{v}_r^*(\mathbf{x}_t,t) = -\mathbf{x}_t - t\,\mathbf{s}_{r,t}(\mathbf{x}_t).$$

Moving the sample and velocity terms to the left-hand side gives:

$$\mathbf{x}_t + (1-t)\mathbf{v}_r^*(\mathbf{x}_t,t) = -t\,\mathbf{s}_{r,t}(\mathbf{x}_t),$$

which, upon dividing both sides by $-t$, gives Eq. (15):

$$\mathbf{s}_{r,t}(\mathbf{x}_t) = -\frac{\mathbf{x}_t + (1-t)\mathbf{v}_r^*(\mathbf{x}_t,t)}{t}.$$

□

Corollary 1 (Velocity Difference as Score Difference). *For $i \in \{1,2\}$, let $p_{i,t}$ be the marginal of $\mathbf{x}_t = (1-t)\mathbf{x}+t\mathbf{z}$ under endpoint law i and independent Gaussian noise. Define $\mathbf{v}_i^*(\mathbf{y},t) := \mathbb{E}_i[\mathbf{z}-\mathbf{x} \mid \mathbf{x}_t = \mathbf{y}, t]$ and $\mathbf{s}_{i,t}(\mathbf{y}) := \nabla_{\mathbf{y}} \log p_{i,t}(\mathbf{y})$. At the same state and time,*

$$\mathbf{v}_1^*(\mathbf{x}_t,t) - \mathbf{v}_2^*(\mathbf{x}_t,t) = -\frac{t}{1-t}\left[\mathbf{s}_{1,t}(\mathbf{x}_t) - \mathbf{s}_{2,t}(\mathbf{x}_t)\right]. \tag{20}$$

Proof. Applying Proposition 1 to both velocity fields:

$$\mathbf{s}_{1,t}(\mathbf{x}_t) = -\frac{\mathbf{x}_t + (1-t)\mathbf{v}_1^*(\mathbf{x}_t,t)}{t}, \tag{21}$$

$$\mathbf{s}_{2,t}(\mathbf{x}_t) = -\frac{\mathbf{x}_t + (1-t)\mathbf{v}_2^*(\mathbf{x}_t,t)}{t}. \tag{22}$$

Subtracting Eq. (22) from Eq. (21) cancels the common state term at the same time:

$$\begin{aligned}\mathbf{s}_{1,t}(\mathbf{x}_t) - \mathbf{s}_{2,t}(\mathbf{x}_t) &= -\frac{1}{t}\left[(\mathbf{x}_t + (1-t)\mathbf{v}_1^*) - (\mathbf{x}_t + (1-t)\mathbf{v}_2^*)\right] \\ &= -\frac{1-t}{t}\left[\mathbf{v}_1^*(\mathbf{x}_t,t) - \mathbf{v}_2^*(\mathbf{x}_t,t)\right].\end{aligned} \tag{23}$$

Rearranging yields Eq. (20) for the difference between the two velocity fields. □

B.3 Frozen-Target Velocity Correction

For $i \in \{r,\theta\}$ and any state $\mathbf{y}$, applying the OT conversion to the estimated velocity field defines

$$\widehat{\mathbf{s}}_{i,t'}(\mathbf{y} \mid \mathbf{c}) := -\frac{\mathbf{y} + (1-t')\widehat{\mathbf{v}}_{i,t'}(\mathbf{y} \mid \mathbf{c})}{t'}. \tag{24}$$

Evaluating both fields at $\mathbf{y} = \mathbf{x}_{\theta,t'}^{\mathrm{sg}}$ and subtracting gives Eq. (7). Interpreting this identity between estimates as a difference of true scores requires calibrated real and fake velocities.

Let $J_1 := J_\theta \boldsymbol{F}_\theta(\mathbf{z},1,\mathbf{c})$. Because the entire target in Eq. (8) is detached, its parameter gradient is

$$\nabla_\theta \mathcal{L}_{\mathrm{score}} = 2\mathbb{E}\left[J_1^\top \,\mathrm{sg}\left(\widetilde{\mathbf{d}}_{t'}\right)\right]. \tag{25}$$

Only the prediction call contributes a derivative; the correction and the current prediction inside the target are held fixed for this update. This product is obtained by ordinary backpropagation and does not require explicitly forming a Jacobian. Shared parameters are still changed by the update, so the statement concerns gradient paths within the loss, not independent branch parameters.

The raw difference $\widehat{\mathbf{d}}_{t'}$ is evaluated at $\mathbf{x}_{\theta,t'}^{\mathrm{sg}}$; $\widetilde{\mathbf{d}}_{t'}$ denotes the correction after the chosen weighting and normalization. If no such processing is used, $\widetilde{\mathbf{d}}_{t'} = \widehat{\mathbf{d}}_{t'}(\mathbf{x}_{\theta,t'}^{\mathrm{sg}} \mid \mathbf{c})$. Equation (25) holds for either choice. The score–velocity conversion motivates the correction but does not, by itself, identify every weighted or normalized update with an exact KL gradient.

Condition shifting with scale and bias. The negated condition illustrated in Section 3.1 is the special case $\mathbf{A} = -\mathbf{I}$, $\mathbf{b} = \mathbf{0}$ of the following affine transformation of the text embedding:

$$\mathbf{c}_{\mathrm{shift}} = \mathbf{A}\mathbf{c} + \mathbf{b}. \tag{26}$$

The condition-construction ablations retain this more general parameterization. Invertibility of $\mathbf{A}$ preserves the input information, but does not ensure that the shifted condition is well calibrated by a pretrained model on the fake-trajectory estimation task.

B.4 Endpoint Formulation for the Loss Ablations

The reported loss-removal and weighting ablations use the endpoint formulation below. They specify the supervised and mixed endpoint losses compared in Section 4.4. As in the main text, r identifies the original-condition prediction and θ the shifted-condition prediction. Here, the shifted field is learned from the time-dependent endpoint samples below; $p^{\mathrm{end}}_{\theta,t}$ denotes their re-noised marginal at the reused time t under the same noise schedule as the reference samples.

The endpoint construction forms a detached prediction from a partially noised reference sample, rather than the pure-noise construction in Eq. (5) used at inference:

$$\mathbf{x}^{\mathrm{end}}_{\theta}(t) := \mathrm{sg}[\mathbf{x}_t - t\boldsymbol{F}_\theta(\mathbf{x}_t, t, \mathbf{c})]\,. \tag{27}$$

Reusing the same t, let $\mathbf{x}^{\mathrm{end}}_{\theta,t} = (1-t)\mathbf{x}^{\mathrm{end}}_{\theta}(t) + t\boldsymbol{\epsilon}'$ with independent $\boldsymbol{\epsilon}' \sim p_z$. Under $\mathbf{c}_{\mathrm{shift}} = \mathbf{A}\mathbf{c}+\mathbf{b}$, the auxiliary loss supervises the fake velocity at this re-noised endpoint:

$$\mathcal{L}^{\mathrm{end}}_{\mathrm{fake}} := \mathbb{E}\Big[\big\|\boldsymbol{F}_\theta(\mathbf{x}^{\mathrm{end}}_{\theta,t}, t, \mathbf{c}_{\mathrm{shift}}) - (\boldsymbol{\epsilon}' - \mathbf{x}^{\mathrm{end}}_{\theta}(t))\big\|_2^2\Big]\,. \tag{28}$$

The superscript end distinguishes this auxiliary loss from Eq. (6). For endpoint regression, the branch evaluations at the same real-data path point are

$$\begin{aligned}\widehat{\mathbf{v}}_{r,t} &:= \boldsymbol{F}_\theta(\mathbf{x}_t, t, \mathbf{c}),\\ \widehat{\mathbf{v}}_{\theta,t} &:= \mathrm{sg}[\boldsymbol{F}_\theta(\mathbf{x}_t, t, \mathbf{c}_{\mathrm{shift}})]\,.\end{aligned} \tag{29}$$

Using the endpoint predictor in Eq. (13), the mixed target and loss are

$$\mathbf{T}_{\mathrm{mix}} := (1-\lambda)\mathbf{x} + \lambda \boldsymbol{f}^{\mathbf{x}}(\widehat{\mathbf{v}}_{\theta,t}, \mathbf{x}_t, t), \qquad \lambda \in [0,1], \tag{30}$$

$$\mathcal{L}_{\mathrm{mix}} := \mathbb{E}\left[\frac{1}{\omega(t)}\,\|\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_\theta, \mathbf{x}_t, t) - \mathbf{T}_{\mathrm{mix}}\|_2^2\right], \qquad \omega(t) = \frac{t}{1-t}. \tag{31}$$

The component losses and their gradient-equivalent combination are

$$\mathcal{L}_{\mathrm{sup}} := \mathbb{E}\left[\frac{1}{\omega(t)}\,\|\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_\theta, \mathbf{x}_t, t) - \mathbf{x}\|_2^2\right], \tag{32}$$

$$\mathcal{L}_{\mathrm{cons}} := \mathbb{E}\left[\frac{1}{\omega(t)}\,\|\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_\theta, \mathbf{x}_t, t) - \boldsymbol{f}^{\mathbf{x}}(\widehat{\mathbf{v}}_{\theta,t}, \mathbf{x}_t, t)\|_2^2\right], \tag{33}$$

$$\mathcal{G}_{\mathrm{APEX}} := (1-\lambda)\mathcal{L}_{\mathrm{sup}} + \lambda\mathcal{L}_{\mathrm{cons}}. \tag{34}$$

The loss-weight ablation varies the outer coefficients in

$$\mathcal{L}_{\mathrm{end}} := \lambda_p\mathcal{L}^{\mathrm{end}}_{\mathrm{fake}} + \lambda_r\mathcal{L}_{\mathrm{sup}} + \lambda_m\mathcal{L}_{\mathrm{mix}}, \qquad \lambda_p, \lambda_r, \lambda_m \geq 0. \tag{35}$$

These coefficients and the inner mixing ratio λ are distinct from λ_{score} in Eq. (9). The endpoint formulation uses $\lambda = 0.5$ and $\lambda_p = \lambda_r = \lambda_m = 1$ unless varied in the corresponding ablation, which changes only the stated loss weights for that comparison.

B.5 Endpoint–Velocity Equivalence

We prove that the endpoint-space MSE and velocity-space MSE are equivalent up to a scalar factor t^2. Comparing weighted losses therefore requires accounting for this time-dependent factor.

Proposition 2 (Endpoint–Velocity Equivalence for Supervised FM). *Let $\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}, \mathbf{x}_t, t) := \mathbf{x}_t - t\,\boldsymbol{F}$ be the endpoint predictor defined in Eq. (13), and let $\mathbf{u}_r(\mathbf{x}, \mathbf{z}) = \mathbf{z} - \mathbf{x}$ be the target velocity under the OT path. Then for any velocity estimate $\boldsymbol{F}_\theta$, we have*

$$\big\|\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_\theta, \mathbf{x}_t, t) - \mathbf{x}\big\|_2^2 = t^2\,\big\|\boldsymbol{F}_\theta - \mathbf{u}_r(\mathbf{x}, \mathbf{z})\big\|_2^2. \tag{36}$$

Proof. **Step 1: Expand the endpoint predictor.** The endpoint predictor in Eq. (13) subtracts the predicted displacement from the noised sample:

$$\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_\theta, \mathbf{x}_t, t) = \mathbf{x}_t - t\,\boldsymbol{F}_\theta. \tag{37}$$

Step 2: Compute the squared error. The LHS of Eq. (36) is

$$\begin{aligned}\big\|\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_\theta, \mathbf{x}_t, t) - \mathbf{x}\big\|_2^2 &= \big\|(\mathbf{x}_t - t\,\boldsymbol{F}_\theta) - \mathbf{x}\big\|_2^2\\ &= \big\|(\mathbf{x}_t - \mathbf{x}) - t\,\boldsymbol{F}_\theta\big\|_2^2.\end{aligned} \tag{38}$$

Step 3: Use the OT path identity. Under the OT path $\mathbf{x}_t = t\mathbf{z} + (1-t)\mathbf{x}$ from Eq. (10), we compute the difference $\mathbf{x}_t - \mathbf{x}$ directly from the linear interpolation:

$$\begin{aligned}\mathbf{x}_t - \mathbf{x} &= \big[t\mathbf{z} + (1-t)\mathbf{x}\big] - \mathbf{x} \quad \text{(substituting the OT path)}\\ &= t\mathbf{z} + (1-t)\mathbf{x} - \mathbf{x}\\ &= t\mathbf{z} + (1-t)\mathbf{x} - 1\cdot\mathbf{x} \quad \text{(writing } \mathbf{x} = 1\cdot\mathbf{x})\\ &= t\mathbf{z} + \mathbf{x} - t\mathbf{x} - \mathbf{x} \quad \text{(expanding } (1-t)\mathbf{x})\\ &= t\mathbf{z} - t\mathbf{x} \quad \text{(canceling } \mathbf{x})\\ &= t(\mathbf{z} - \mathbf{x}). \quad \text{(factoring out } t)\end{aligned} \tag{39}$$

Recall that under the OT path, the target velocity is defined as $\mathbf{u}_r(\mathbf{x}, \mathbf{z}) := \mathbf{z} - \mathbf{x}$, which is the instantaneous rate of change from data $\mathbf{x}$ to noise $\mathbf{z}$. Therefore, we obtain the key identity:

$$\mathbf{x}_t - \mathbf{x} = t\,\mathbf{u}_r(\mathbf{x}, \mathbf{z}). \tag{40}$$

The target velocity is constant along this path because the interpolation is linear in time.

Step 4: Substitute and simplify. Substituting Eq. (40) into Eq. (38):

$$\begin{aligned}\big\|(\mathbf{x}_t - \mathbf{x}) - t\,\boldsymbol{F}_\theta\big\|_2^2 &= \big\|t\,\mathbf{u}_r(\mathbf{x}, \mathbf{z}) - t\,\boldsymbol{F}_\theta\big\|_2^2 \quad \text{(using Eq. (40))}\\ &= \big\|t\,(\mathbf{u}_r(\mathbf{x}, \mathbf{z}) - \boldsymbol{F}_\theta)\big\|_2^2 \quad \text{(factoring out } t)\\ &= t^2\,\big\|\mathbf{u}_r(\mathbf{x}, \mathbf{z}) - \boldsymbol{F}_\theta\big\|_2^2, \quad \text{(using } \|c\mathbf{v}\|_2^2 = c^2\|\mathbf{v}\|_2^2)\end{aligned} \tag{41}$$

which proves Eq. (36). The final step uses the homogeneity property of the squared ℓ_2 norm.

□

Proposition 3 (Endpoint–Velocity Equivalence for Fake Alignment) . *For the fake-alignment term, let* $\widehat{\mathbf{v}}_{\theta,t}(\mathbf{x}_t \mid \mathbf{c}) := \mathrm{sg}(\boldsymbol{F}_\theta(\mathbf{x}_t, t, \mathbf{c}_{\text{shift}}))$ *be the fake velocity evaluated with the same online network* $\boldsymbol{F}_\theta$ *under the shifted condition* $\mathbf{c}_{\text{shift}}$ *with stop-gradient applied. Then*

$$\big\|\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_\theta, \mathbf{x}_t, t) - \boldsymbol{f}^{\mathbf{x}}(\widehat{\mathbf{v}}_{\theta,t}, \mathbf{x}_t, t)\big\|_2^2 = t^2\,\big\|\boldsymbol{F}_\theta - \widehat{\mathbf{v}}_{\theta,t}(\mathbf{x}_t \mid \mathbf{c})\big\|_2^2. \tag{42}$$

Proof. **Step 1: Expand both endpoint predictors.** By definition,

$$\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_\theta, \mathbf{x}_t, t) = \mathbf{x}_t - t\,\boldsymbol{F}_\theta, \tag{43}$$

$$\boldsymbol{f}^{\mathbf{x}}(\widehat{\mathbf{v}}_{\theta,t}, \mathbf{x}_t, t) = \mathbf{x}_t - t\,\widehat{\mathbf{v}}_{\theta,t}(\mathbf{x}_t \mid \mathbf{c}). \tag{44}$$

Step 2: Compute the difference.

$$\begin{aligned}\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_\theta, \mathbf{x}_t, t) - \boldsymbol{f}^{\mathbf{x}}(\widehat{\mathbf{v}}_{\theta,t}, \mathbf{x}_t, t) &= \big[\mathbf{x}_t - t\,\boldsymbol{F}_\theta\big] - \big[\mathbf{x}_t - t\,\widehat{\mathbf{v}}_{\theta,t}\big]\\ &= \mathbf{x}_t - t\,\boldsymbol{F}_\theta - \mathbf{x}_t + t\,\widehat{\mathbf{v}}_{\theta,t}\\ &= t\,\widehat{\mathbf{v}}_{\theta,t} - t\,\boldsymbol{F}_\theta\\ &= t\,(\widehat{\mathbf{v}}_{\theta,t} - \boldsymbol{F}_\theta).\end{aligned} \tag{45}$$

Step 3: Square the norm.

$$\begin{aligned}\big\|\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_\theta, \mathbf{x}_t, t) - \boldsymbol{f}^{\mathbf{x}}(\widehat{\mathbf{v}}_{\theta,t}, \mathbf{x}_t, t)\big\|_2^2 &= \big\|t\,(\widehat{\mathbf{v}}_{\theta,t} - \boldsymbol{F}_\theta)\big\|_2^2\\ &= t^2\,\big\|\widehat{\mathbf{v}}_{\theta,t} - \boldsymbol{F}_\theta\big\|_2^2\\ &= t^2\,\big\|\boldsymbol{F}_\theta - \widehat{\mathbf{v}}_{\theta,t}(\mathbf{x}_t \mid \mathbf{c})\big\|_2^2,\end{aligned} \tag{46}$$

which proves Eq. (42) for both endpoint predictions at the same state and time. □

B.6 Gradient Equivalence of Alternative Loss

We now show that the gradient of the mixed endpoint loss $\mathcal{L}_{\text{mix}}$ is equal to the gradient of the alternative loss $\mathcal{G}_{\text{APEX}}$. The equivalence requires that the mixed target is detached when differentiating the loss with respect to the shared parameters.

Theorem 1 (Gradient Equivalence) . *Let $\mathcal{L}_{mix}(\theta)$ and $\mathcal{G}_{\text{APEX}}(\theta)$ be defined as in Eq.* (31) *and Eq.* (34)*, respectively. Then, for any parameter θ, differentiating the two objectives gives*

$$\nabla_\theta \mathcal{L}_{mix}(\theta) = \nabla_\theta \mathcal{G}_{\text{APEX}}(\theta). \tag{47}$$

Proof. For notational simplicity, we work with one training pair and omit the expectation, while retaining the time weight $1/\omega(t)$ in each loss. Write $\boldsymbol{F}_\theta \equiv \boldsymbol{F}_\theta(\mathbf{x}_t, t, \mathbf{c})$, $J_t := \partial \boldsymbol{F}_\theta / \partial\theta \in \mathbb{R}^{d\times P}$, and $\widehat{\mathbf{v}}_{\theta,t} \equiv \widehat{\mathbf{v}}_{\theta,t}(\mathbf{x}_t \mid \mathbf{c})$. The mixed target $\mathbf{T}_{\text{mix}}$ includes the sampled clean data $\mathbf{x}$; all its inputs are held fixed in the gradient while differentiating the original-condition prediction.

Part A: Gradient of the mixed endpoint loss. Step A1: Write the mixed endpoint loss. From Eq. (31), the mixed endpoint loss is written in terms of its detached target as

$$\mathcal{L}_{\text{mix}}(\theta) = \frac{1}{\omega(t)} \left\| \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_\theta, \mathbf{x}_t, t) - \mathbf{T}_{\text{mix}} \right\|_2^2, \tag{48}$$

where the mixed target is defined in Eq. (30) as the following weighted sum:

$$\mathbf{T}_{\text{mix}} = (1-\lambda)\,\mathbf{x} + \lambda\, \boldsymbol{f}^{\mathbf{x}}(\widehat{\mathbf{v}}_{\theta,t}, \mathbf{x}_t, t). \tag{49}$$

Step A2: Expand the endpoint predictors. Using the definition $\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}, \mathbf{x}_t, t) = \mathbf{x}_t - t\,\boldsymbol{F}$ from Eq. (13), the original and shifted predictions take the following forms:

$$\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_\theta, \mathbf{x}_t, t) = \mathbf{x}_t - t\,\boldsymbol{F}_\theta, \tag{50}$$

$$\boldsymbol{f}^{\mathbf{x}}(\widehat{\mathbf{v}}_{\theta,t}, \mathbf{x}_t, t) = \mathbf{x}_t - t\,\widehat{\mathbf{v}}_{\theta,t}. \tag{51}$$

Step A3: Substitute into the mixed target. Substituting Eq. (51) into Eq. (49):

$$\begin{aligned} \mathbf{T}_{\text{mix}} &= (1-\lambda)\,\mathbf{x} + \lambda\,(\mathbf{x}_t - t\,\widehat{\mathbf{v}}_{\theta,t}) \\ &= (1-\lambda)\,\mathbf{x} + \lambda\,\mathbf{x}_t - \lambda t\,\widehat{\mathbf{v}}_{\theta,t}. \end{aligned} \tag{52}$$

Step A4: Compute the error term $\boldsymbol{\delta}$. Define the error as

$$\boldsymbol{\delta} := \boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_\theta, \mathbf{x}_t, t) - \mathbf{T}_{\text{mix}}. \tag{53}$$

Substituting Eq. (50) and Eq. (52) into this definition of the endpoint residual gives

$$\begin{aligned} \boldsymbol{\delta} &= (\mathbf{x}_t - t\,\boldsymbol{F}_\theta) - \big[(1-\lambda)\,\mathbf{x} + \lambda\,\mathbf{x}_t - \lambda t\,\widehat{\mathbf{v}}_{\theta,t}\big] \\ &= \mathbf{x}_t - t\,\boldsymbol{F}_\theta - (1-\lambda)\,\mathbf{x} - \lambda\,\mathbf{x}_t + \lambda t\,\widehat{\mathbf{v}}_{\theta,t} \\ &= \mathbf{x}_t(1-\lambda) - (1-\lambda)\,\mathbf{x} - t\,\boldsymbol{F}_\theta + \lambda t\,\widehat{\mathbf{v}}_{\theta,t} \\ &= (1-\lambda)(\mathbf{x}_t - \mathbf{x}) - t\,\boldsymbol{F}_\theta + \lambda t\,\widehat{\mathbf{v}}_{\theta,t}. \end{aligned} \tag{54}$$

Step A5: Apply the OT path identity. From Eq. (40) (proven in Section B.5), we have

$$\mathbf{x}_t - \mathbf{x} = t\,\mathbf{u}_r, \quad \text{where } \mathbf{u}_r = \mathbf{z} - \mathbf{x}. \tag{55}$$

Substituting into Eq. (54) expresses the endpoint residual through the velocity predictions:

$$\begin{aligned} \boldsymbol{\delta} &= (1-\lambda)\,t\,\mathbf{u}_r - t\,\boldsymbol{F}_\theta + \lambda t\,\widehat{\mathbf{v}}_{\theta,t} \\ &= t\big[(1-\lambda)\,\mathbf{u}_r + \lambda\,\widehat{\mathbf{v}}_{\theta,t} - \boldsymbol{F}_\theta\big]. \end{aligned} \tag{56}$$

Step A6: Compute the gradient using the chain rule. The time weight is independent of θ. For the vector residual $\boldsymbol{\delta} \in \mathbb{R}^d$, the chain rule gives a parameter vector:

$$\nabla_\theta \mathcal{L}_{\text{mix}} = \frac{2}{\omega(t)} \left(\frac{\partial \boldsymbol{\delta}}{\partial \theta}\right)^\top \boldsymbol{\delta}. \tag{57}$$

The sampled $\mathbf{x}, \mathbf{z}, t$ and the detached shifted prediction are fixed, so

$$\frac{\partial \boldsymbol{\delta}}{\partial \theta} = -t J_t. \tag{58}$$

Step A7: Substitute the endpoint residual. Using Eqs. (56) and (58),

$$\begin{aligned} \nabla_\theta \mathcal{L}_{\text{mix}} &= \frac{2t^2}{\omega(t)} J_t^\top \big[\boldsymbol{F}_\theta - (1-\lambda)\mathbf{u}_r - \lambda \widehat{\mathbf{v}}_{\theta,t}\big] \\ &= 2t(1-t) J_t^\top \big[\boldsymbol{F}_\theta - (1-\lambda)\mathbf{u}_r - \lambda \widehat{\mathbf{v}}_{\theta,t}\big], \end{aligned} \tag{59}$$

where $t^2/\omega(t) = t(1-t)$, as in Eq. (64), for the sampled time in this expectation.

Step A8: Separate the two residuals. By linearity of multiplication by $J_t^\top$,

$$\nabla_\theta \mathcal{L}_{\text{mix}} = 2t(1-t) J_t^\top \big[(1-\lambda)(\boldsymbol{F}_\theta - \mathbf{u}_r) + \lambda(\boldsymbol{F}_\theta - \widehat{\mathbf{v}}_{\theta,t})\big]. \tag{60}$$

Part B: Gradient of the alternative loss. **Step B1: Write the alternative loss.** From Eq. (34), the alternative loss is the weighted combination of supervision and consistency:

$$\mathcal{G}_{\text{APEX}}(\theta) = (1-\lambda)\,\mathcal{L}_{\text{sup}}(\theta) + \lambda\,\mathcal{L}_{\text{cons}}(\theta), \tag{61}$$

where $\mathcal{L}_{\text{sup}}$ and $\mathcal{L}_{\text{cons}}$ are defined in Eq. (32) and Eq. (33).

Step B2: Apply the endpoint-velocity equivalence. By Proposition 2, we have

$$\mathcal{L}_{\text{sup}}(\theta) = \frac{1}{\omega(t)}\left\|\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_\theta, \mathbf{x}_t, t) - \mathbf{x}\right\|_2^2 = \frac{t^2}{\omega(t)}\left\|\boldsymbol{F}_\theta - \mathbf{u}_r\right\|_2^2. \tag{62}$$

By Proposition 3, the fake-alignment term can be written as a velocity error:

$$\mathcal{L}_{\text{cons}}(\theta) = \frac{1}{\omega(t)}\left\|\boldsymbol{f}^{\mathbf{x}}(\boldsymbol{F}_\theta, \mathbf{x}_t, t) - \boldsymbol{f}^{\mathbf{x}}(\widehat{\mathbf{v}}_{\theta,t}, \mathbf{x}_t, t)\right\|_2^2 = \frac{t^2}{\omega(t)}\left\|\boldsymbol{F}_\theta - \widehat{\mathbf{v}}_{\theta,t}\right\|_2^2. \tag{63}$$

Step B3: Compute the gradients of $\mathcal{L}_{\text{sup}}$ and $\mathcal{L}_{\text{cons}}$. Both losses carry the time weight $1/\omega(t)$ of Eqs. (32)–(33) in addition to the t^2 factor above. Since $\omega(t) = t/(1-t)$ depends only on the sampled time and not on $\mathbf{x}_t$, the two factors combine into a single per-sample weight

$$\frac{1}{\omega(t)} \cdot t^2 = \frac{1-t}{t} \cdot t^2 = t\,(1-t), \tag{64}$$

which we carry explicitly from here on. Applying the same chain rule to both residuals gives

$$\nabla_\theta \mathcal{L}_{\text{sup}} = 2t(1-t) J_t^\top (\boldsymbol{F}_\theta - \mathbf{u}_r), \tag{65}$$

$$\nabla_\theta \mathcal{L}_{\text{cons}} = 2t(1-t) J_t^\top (\boldsymbol{F}_\theta - \widehat{\mathbf{v}}_{\theta,t}). \tag{66}$$

The stop-gradient on $\widehat{\mathbf{v}}_{\theta,t}$ is essential: it removes its derivative even though both calls share parameters through the same network weights used by the generator.

Step B4: Combine the gradients. Substituting these expressions into Eq. (61) yields

$$\begin{aligned}\nabla_\theta \mathcal{G}_{\text{APEX}} &= (1-\lambda)\nabla_\theta \mathcal{L}_{\text{sup}} + \lambda \nabla_\theta \mathcal{L}_{\text{cons}} \\ &= 2t(1-t) J_t^\top \left[(1-\lambda)(\boldsymbol{F}_\theta - \mathbf{u}_r) + \lambda(\boldsymbol{F}_\theta - \widehat{\mathbf{v}}_{\theta,t})\right].\end{aligned} \tag{67}$$

Part C: Conclusion. Comparing Eq. (60) and Eq. (67), we see that the gradients match:

$$\nabla_\theta\, \mathcal{L}_{\text{mix}}(\theta) = \nabla_\theta\, \mathcal{G}_{\text{APEX}}(\theta). \tag{68}$$

The equality holds for each sampled pair. Taking expectations under the same sampling law proves Theorem 1, assuming differentiation and expectation can be interchanged. □

B.7 Score-Difference Form of the Endpoint Surrogate Gradient

All expectations below use real-data pairs $(\mathbf{x}, \mathbf{c})$, independent noise $\mathbf{z} \sim p_z$, and the sampled time $t \in (0,1)$; $\mathbf{x}_t = (1-t)\mathbf{x} + t\mathbf{z}$. The shifted-condition prediction is detached within the endpoint loss, so only the original-condition prediction contributes its parameter derivative.

Proposition 4 (Score-Difference Form) . *Let $\widehat{\mathbf{v}}_{r,t}$ and $\widehat{\mathbf{v}}_{\theta,t}$ be the original- and shifted-condition predictions from Appendix B.4, evaluated at $(\mathbf{x}_t, \mathbf{c})$. Define their induced fields by $\widehat{\mathbf{s}}_{i,t} := -[\mathbf{x}_t + (1-t)\widehat{\mathbf{v}}_{i,t}]/t$ for $i \in \{r, \theta\}$, and let $\mathbf{s}_{r,t} = \nabla_{\mathbf{x}_t} \log p_{r,t}(\mathbf{x}_t \mid \mathbf{c})$ be the exact noised-data score. The mixed target field uses the same mixing weight:*

$$\widehat{\mathbf{s}}_{\text{mix},t} := (1-\lambda)\mathbf{s}_{r,t} + \lambda \widehat{\mathbf{s}}_{\theta,t}. \tag{69}$$

For $\omega(t) = t/(1-t)$ and $J_t := J_\theta \boldsymbol{F}_\theta(\mathbf{x}_t, t, \mathbf{c})$, the frozen-target gradient is

$$\nabla_\theta \mathcal{G}_{\text{APEX}} = -2\mathbb{E}\left[t^2 J_t^\top \operatorname{sg}(\widehat{\mathbf{s}}_{r,t} - \widehat{\mathbf{s}}_{\text{mix},t})\right]. \tag{70}$$

The coefficient $-2t^2$ depends on time, but is independent of the sample at a fixed t.

Proof. Because J_t depends on the sampled pair only through $(\mathbf{x}_t, t, \mathbf{c})$, conditional expectation replaces the supervised target $\mathbf{u}_r = \mathbf{z} - \mathbf{x}$ with the true data velocity $\mathbf{v}_r^*(\mathbf{x}_t, t, \mathbf{c})$. The field entering the gradient of $\mathcal{G}_{\text{APEX}} = (1-\lambda)\mathcal{L}_{\text{sup}} + \lambda\mathcal{L}_{\text{cons}}$ is therefore

$$(1-\lambda)(\widehat{\mathbf{v}}_{r,t} - \mathbf{v}_r^*) + \lambda(\widehat{\mathbf{v}}_{r,t} - \widehat{\mathbf{v}}_{\theta,t}). \tag{71}$$

The OT conversion and the definitions of the induced fields give

$$\widehat{\mathbf{v}}_{r,t} - \mathbf{v}_r^* = -\frac{t}{1-t}(\widehat{\mathbf{s}}_{r,t} - \mathbf{s}_{r,t}), \tag{72}$$

$$\widehat{\mathbf{v}}_{r,t} - \widehat{\mathbf{v}}_{\theta,t} = -\frac{t}{1-t}(\widehat{\mathbf{s}}_{r,t} - \widehat{\mathbf{s}}_{\theta,t}). \tag{73}$$

Combining these identities expresses the velocity residual in score coordinates:

$$(1-\lambda)(\widehat{\mathbf{v}}_{r,t} - \mathbf{v}_r^*) + \lambda(\widehat{\mathbf{v}}_{r,t} - \widehat{\mathbf{v}}_{\theta,t}) = -\frac{t}{1-t}(\widehat{\mathbf{s}}_{r,t} - \widehat{\mathbf{s}}_{\mathrm{mix},t}). \tag{74}$$

Substituting this identity into the weighted endpoint gradient yields

$$\begin{aligned}\nabla_\theta \mathcal{G}_{\mathrm{APEX}} &= \mathbb{E}\left[\frac{2t^2}{\omega(t)} J_t^\top \,\mathrm{sg}((1-\lambda)(\widehat{\mathbf{v}}_{r,t} - \mathbf{v}_r^*) + \lambda(\widehat{\mathbf{v}}_{r,t} - \widehat{\mathbf{v}}_{\theta,t}))\right] \\ &= -2\mathbb{E}\left[t^2 J_t^\top \,\mathrm{sg}(\widehat{\mathbf{s}}_{r,t} - \widehat{\mathbf{s}}_{\mathrm{mix},t})\right], \end{aligned} \tag{75}$$

where $t^2/\omega(t) = t(1-t)$. All time-dependent factors remain inside the expectation over t. □

Remark 1 (Scope of the gradient identity) . *This identity rewrites the frozen-target gradient of the endpoint objective. It does not require the network-induced fields $\widehat{\mathbf{s}}_{r,t}$ and $\widehat{\mathbf{s}}_{\theta,t}$ to equal true distribution scores, and it does not identify the loss with a divergence between generated and data distributions. The coefficient is constant across samples only at a fixed time; the residual and network Jacobian still vary across samples.*

Remark 2 (When the mixed field has a density interpretation) . *For each fixed t, $p^{\mathrm{end}}_{\theta,t}$ is the law of $(1-t)\mathbf{x}^{\mathrm{end}}_\theta(t) + t\boldsymbol{\epsilon}'$ from Appendix B.4 , with independent Gaussian $\boldsymbol{\epsilon}'$. The clean endpoint law itself depends on t; the score conversion applies at each fixed time. If the shifted-condition field is calibrated so that $\widehat{\mathbf{s}}_{\theta,t} = \nabla \log p^{\mathrm{end}}_{\theta,t}$, then $\widehat{\mathbf{s}}_{\mathrm{mix},t}$ is the score of*

$$p_{\mathrm{mix},t}(\mathbf{x}_t \mid \mathbf{c}) \propto p_{r,t}(\mathbf{x}_t \mid \mathbf{c})^{1-\lambda} \left[p^{\mathrm{end}}_{\theta,t}(\mathbf{x}_t \mid \mathbf{c})\right]^{\lambda}, \tag{76}$$

provided this expression is normalizable. This is a geometric combination at a fixed time, and need not define a consistent probability path across time. Without the calibration assumption, $\widehat{\mathbf{s}}_{\mathrm{mix},t}$ is an induced target field rather than an asserted density score.

Relation to GAN weighting. For comparison, define the noised generated sample $\mathbf{x}_{\theta,t} := (1-t)\boldsymbol{G}_\theta(\mathbf{z},\mathbf{c}) + t\boldsymbol{\epsilon}'$ and its Jacobian $J_t^G := J_\theta \mathbf{x}_{\theta,t}$. The non-saturating GAN gradient at a fixed noise level can be written with its discriminator-dependent sample weight as

$$\nabla_\theta \mathcal{L}_{\text{NS-GAN}} = \mathbb{E}\left[(J_t^G)^\top \, w_{\mathrm{NS}}(\mathbf{x}_{\theta,t}) \, (\mathbf{s}_{\theta,t}(\mathbf{x}_{\theta,t}) - \mathbf{s}_{r,t}(\mathbf{x}_{\theta,t}))\right], \tag{77}$$

where $w_{\mathrm{NS}} = 1 - D_t^*$ for the optimal discriminator held fixed during the generator update. Unlike this discriminator-dependent weight, the explicit coefficient in Eq. (70) depends only on time. The expectations and Jacobians correspond to their respective training constructions; the comparison concerns how those constructions weight each sample in their gradients.

B.8 GAN Dynamics and Score-Difference Gradients

Let $p_\theta(\mathbf{x})$ and $p_r(\mathbf{x})$ denote the generated and data densities for a fixed condition $\mathbf{c}$, omitted below for brevity, and let $\mathbf{s}(\mathbf{x}) := \nabla_{\mathbf{x}} \log p(\mathbf{x})$ denote a distribution's score. With the optimal discriminator $D^*(\mathbf{x}) = {p_r(\mathbf{x})}/{p_r(\mathbf{x}) + p_\theta(\mathbf{x})}$ held fixed during the generator update (Mohamed & Lakshminarayanan, 2016; Goodfellow et al., 2014), the saturating and non-saturating losses yield

$$\nabla_\theta \mathcal{L}_{\mathrm{GAN}} \propto \mathbb{E}_{\mathbf{x}\sim p_\theta}\left[\left(\frac{\partial \mathbf{x}}{\partial \theta}\right)^\top w(\mathbf{x})\big(\mathbf{s}_\theta(\mathbf{x}) - \mathbf{s}_r(\mathbf{x})\big)\right]. \tag{78}$$

The sample is $\mathbf{x} = \boldsymbol{G}_\theta(\mathbf{z},\mathbf{c})$. The weight is $w(\mathbf{x}) = D^*(\mathbf{x})$ for the saturating loss and $w(\mathbf{x}) = 1 - D^*(\mathbf{x})$ for the non-saturating loss.

C VISUALIZATIONS PART I

This section provides additional qualitative results to complement the quantitative analysis in the main paper using the model configurations indicated in each figure.



Figure 4: **Qualitative comparison at 512×512 for APEX 20B LoRA with 1 NFE.**



Figure 5: **Qualitative comparison at 512×512 for APEX 20B LoRA with 1 NFE.**



Figure 6: **Qualitative comparison at 512×512 for APEX 20B LoRA with 1 NFE.**



Figure 7: **Qualitative comparison at 512×512 for APEX 0.6B LoRA with 1 NFE.**

D VISUALIZATIONS PART II



Figure 8: **Qualitative comparison at 512×512 for APEX 20B LoRA with 1 NFE.**



Figure 9: **Qualitative comparison at 512×512 for APEX 20B LoRA with 1 NFE.**

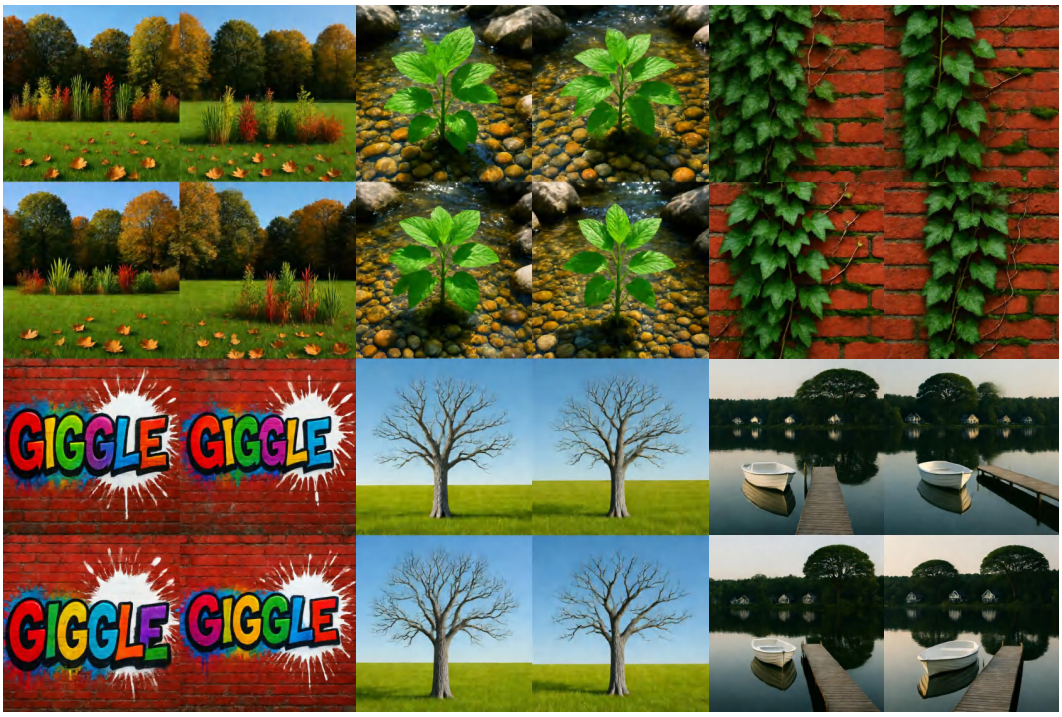

Figure 10: **Qualitative comparison at 512×512 for APEX 20B LoRA with 1 NFE.**



Figure 11: **Qualitative comparison at 512×512 for APEX 20B full-parameter tuning with 1 NFE.**

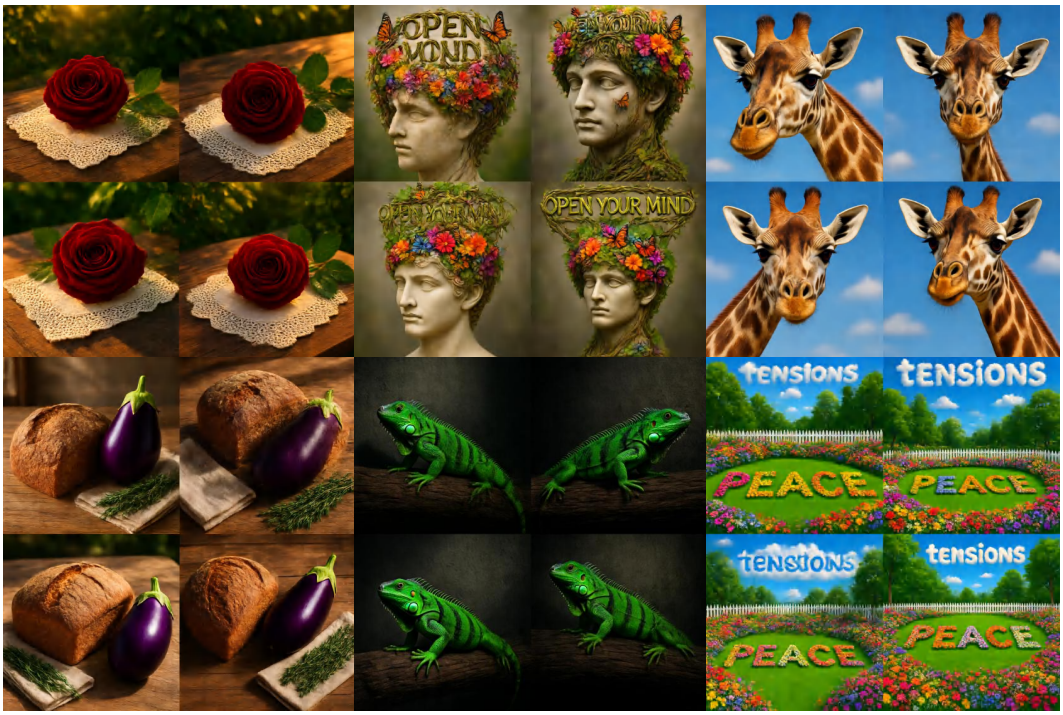

Figure 12: **Qualitative comparison at 512×512 for APEX 20B full-parameter tuning with 1 NFE.**



Figure 13: **Qualitative comparison at 512×512 for APEX 20B full-parameter tuning with 1 NFE.**



Figure 14: **Qualitative comparison at 512×512 for Qwen-Image Lightning LoRA with 1 NFE.**

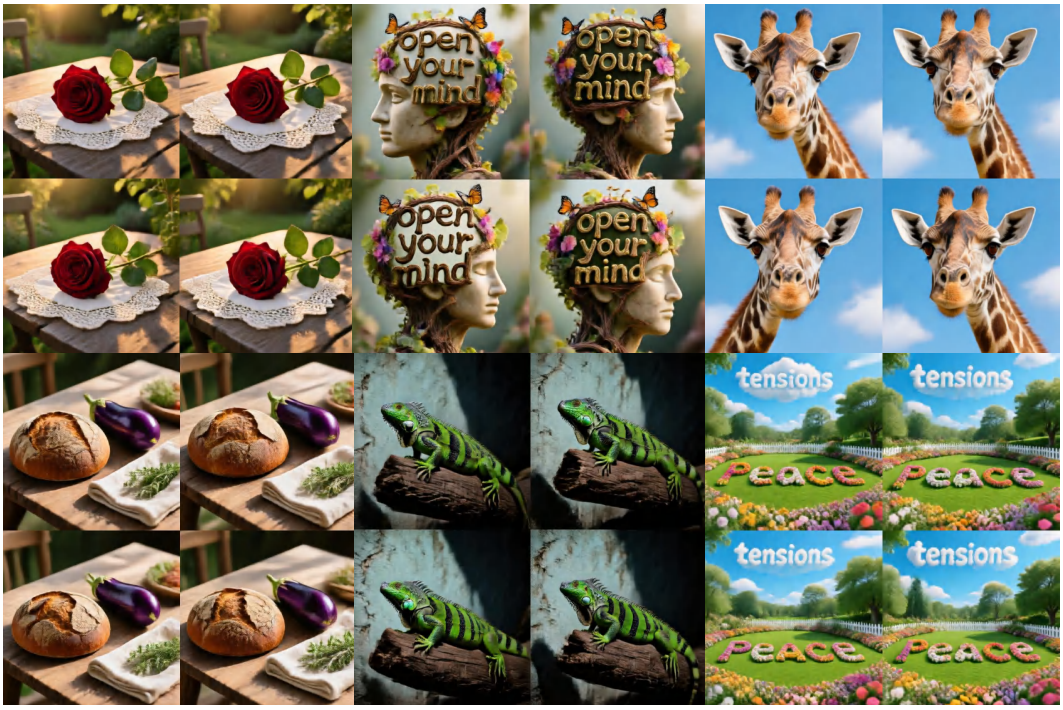

Figure 15: **Qualitative comparison at 512×512 for Qwen-Image Lightning LoRA with 1 NFE.**



Figure 16: **Qualitative comparison at 512×512 for Qwen-Image Lightning LoRA with 1 NFE.**

E VISUALIZATIONS PART III



Figure 17: **Qualitative comparison at 512×512 for APEX 20B full-parameter tuning on the synthetic dataset from 1 NFE to 20 NFE.**

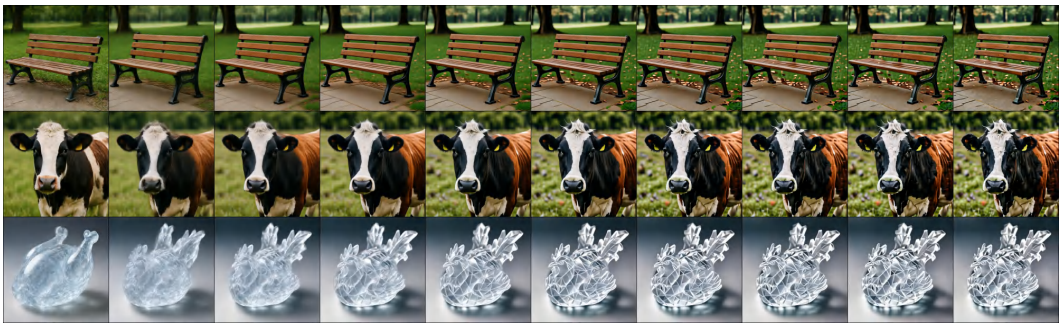

Figure 18: **Qualitative comparison at 512×512 for APEX 20B full-parameter tuning on the BLIP-3o dataset from 1 NFE to 20 NFE.**

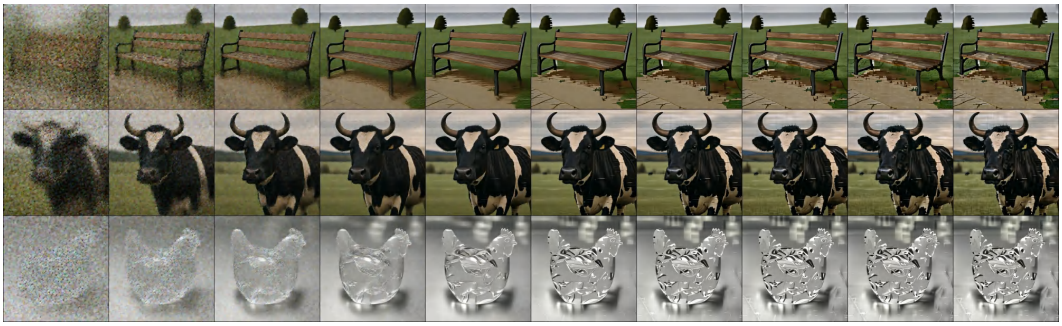

Figure 19: **Qualitative comparison at 512×512 for sCM 20B full-parameter tuning on the BLIP-3o dataset from 1 NFE to 20 NFE.**



Figure 20: **Qualitative comparison at 512×512 for CTM 20B full-parameter tuning on the BLIP-3o dataset from 1 NFE to 20 NFE.**



Figure 21: **Qualitative comparison at 512×512 for MeanFlow 20B full-parameter tuning on the BLIP-3o dataset from 1 NFE to 20 NFE.**

F LIMITATIONS

Our current work primarily focuses on fine-tuning from pretrained backbones rather than training from scratch, and has been mainly validated on large-scale models (0.6B–20B); extending APEX to training from scratch settings or smaller models warrants further study. Additionally, our training datasets remain modest in scale ($\leq$800K images); scaling to larger and more diverse corpora may further improve quality. Moreover, APEX has only been validated on text-to-image generation; extending it to video synthesis, 3D generation, or other continuous modalities is a promising direction but remains unexplored. Our experiments use limited fine-tuning budgets; the behavior of APEX under substantially longer training remains to be studied.

The reported scores follow standard benchmark protocols. Uncertainty across independent training runs and paired significance tests are not reported, so small numerical gaps describe the evaluated checkpoints rather than a quantified stability margin. Conditional diversity, mode coverage, and human preference remain unmeasured; the qualitative grids illustrate individual outputs. The internal correction avoids an external discriminator, but its effect on the diversity loss associated with adversarial one-step training has not been measured through conditional diversity or mode coverage.

The score-difference interpretation in Section 3.2 depends on the real and fake velocity estimates approximating the conditional-mean fields of their noised distributions. The generated endpoints in Eq. (5) use the same pure-noise one-step map as inference, but we do not quantify the residual velocity-estimation error or its effect on the correction. The intermediate-time endpoint construction in Appendix B.4 has an additional training–inference marginal mismatch; the associated proofs apply to the endpoint losses used in the ablations, and do not establish guarantees for the velocity-regression objective in the main text.