A LIGHTWEIGHT VELOCITY CORRECTION FRAMEWORK FOR IMPROVING INSTRUCTION FOLLOWING IN FEW-STEP TEXT-TO-IMAGE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Few-step text-to-image generators produce high-quality images with few network forward passes, yet struggle with complex prompts involving multiple subjects, interactions, spatial relations, counting, and attribute binding. Can these errors be reduced without changing the generator's parameters or increasing its forward-pass count? We propose a lightweight correction framework for a frozen 6B-parameter generator. The generator predicts velocities that determine how its intermediate image representation is updated; a corrector with fewer than 20M parameters learns residual adjustments at selected sampling steps. Because earlier corrections change the states encountered at later steps, our On-Trajectory Flow-Bridge Supervision derives residual targets at the states actually reached during corrected generation. A single corrector is shared across selected steps and runs in parallel with the generator. Only the corrector is trained, requiring neither reward optimization nor re-distillation. To our knowledge, this is the first framework for lightweight velocity-residual adaptation of fully frozen few-step text-to-image generators whose corrector runs in parallel with the generator while preserving its original sampling schedule and forward-pass count. On our target complex-scene distribution, the framework achieves a NetWin of +17.60 percentage points over frozen Z-Image-Turbo in pairwise instruction-following evaluation, with only 26.9 ms (0.84%) additional end-to-end latency per image and no increase in backbone evaluations.

1 INTRODUCTION

Few-step text-to-image generation aims to recover the effectiveness of multi-step denoising with only a limited number of network forward passes. Recent diffusion and flow distillation methods have made substantial progress toward this goal (Kim et al., 2024; Yin et al., 2024; Ren et al., 2024; Zheng et al., 2026b). Models such as Z-Image-Turbo already produce images of high visual quality (Z-Image Team et al., 2025; Jiang et al., 2025), yet their adherence to complex instructions remains limited, particularly for multi-subject interactions, spatial relations, counting, and attribute binding. Thus, visually plausible outputs can still miss important details specified in the prompt.

Existing post-training methods improve instruction following by adapting the generator (Jiang et al., 2024; Black et al., 2024; Prabhudesai et al., 2024; Jiang et al., 2025; 2026). However, adapting a step-distilled generator can disrupt its learned few-step sampling behavior, compromising image quality under the original schedule. To explore an alternative, we draw inspiration from the stronger complex-instruction following of 50-step Z-Image compared with eight-step Z-Image-Turbo on our target distribution (Fig. 1(c)). Comparing their sampling trajectories from shared initial noise reveals substantial differences in update direction over matched noise intervals, despite broadly comparable relative update magnitudes (Fig. 1(a,b)). This suggests intermediate velocities as a potential intervention point: can we correct these velocities to improve few-step instruction following while keeping the generator frozen and its sampling schedule unchanged?

Our key idea is to keep the generator frozen and train a lightweight auxiliary branch to add residual corrections to its predicted latent-space velocities, preserving the sampling schedule and backbone forward-pass count. A step-distilled generator has learned update behavior adapted to its own sam-

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.

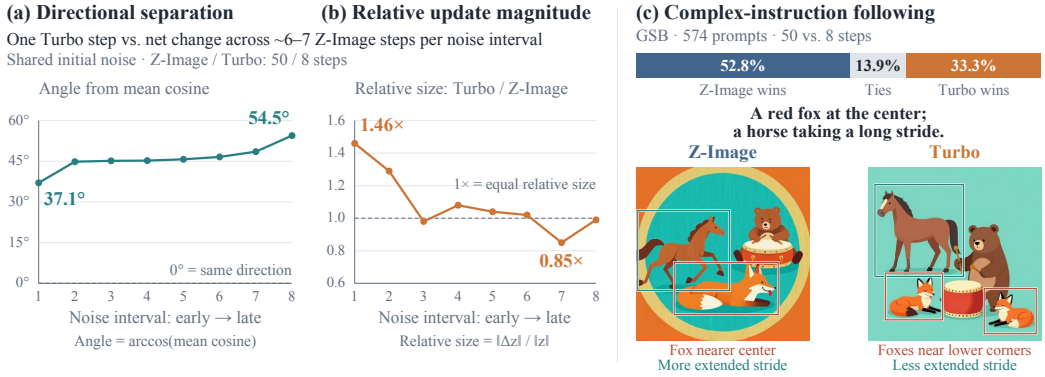


Figure 1: **Few-step and multi-step generation: intermediate updates and instruction following.** **(a)** Intermediate update directions differ substantially, with angles corresponding to mean cosine similarities increasing from $37.1°$ to $54.5°$. **(b)** Relative update magnitudes remain broadly comparable, with Turbo-to-Z-Image ratios ranging from 0.85 to 1.46. **(c)** On 574 complex-scene prompts, 50-step Z-Image outperforms eight-step Z-Image-Turbo by +19.5 pp NetWin in pairwise instruction-following evaluation. The example highlights differences in satisfying the specified fox placement and horse stride.

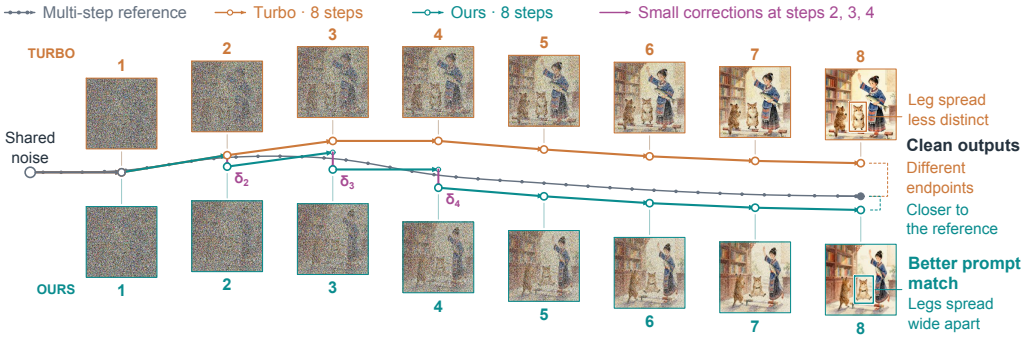


Figure 2: **Sparse velocity correction for few-step generation.** From shared noise, Turbo (orange) and Ours (teal) generate images in eight steps, alongside a multi-step reference (gray). Ours freezes the Turbo backbone and adds velocity residuals $r_{\phi,k}$ only at steps $k \in \{2, 3, 4\}$. Purple arrows δ_k indicate the resulting trajectory displacements; markers link intermediate states to image thumbnails. The highlighted animal in Ours better satisfies "its front and hind legs spread wide apart." Trajectories and endpoint gaps are schematic.

pling schedule, so directly matching a multi-step model's intermediate velocities may not preserve its few-step denoising behavior. We therefore learn residual corrections from target-image supervision along the corrected few-step trajectories. Their effectiveness depends on both prediction accuracy and intervention timing. When a correction is insufficiently accurate, applying it at a late step may leave too few subsequent updates to resolve the resulting deviation. We therefore apply corrections at selected early steps, allowing the remaining frozen updates to accommodate residual inaccuracies and complete denoising.

Realizing this idea requires addressing three challenges: limiting inference overhead, defining suitable residual targets, and learning sufficiently accurate corrections. **Parallel Velocity Residual Correction** addresses the first by conditioning on the current latent and cached previous-step velocity, allowing the corrector to run alongside the current generator evaluation. To define suitable targets, **On-Trajectory Flow-Bridge Supervision** constructs a linear reference bridge from the initial noise to a target latent obtained by offline editing of the generator's output. Because earlier corrections change the states encountered at later steps, it recomputes residual targets at the states actually visited by the corrected sampler, combining reference-velocity correction with trajectory recovery. To improve correction accuracy, **Shared Multi-Step Learning** jointly trains a single noise-conditioned corrector across multiple selected early steps using local supervision, without

backpropagating through preceding sampling updates. Figure 2 illustrates the resulting improvement within the original eight-step schedule: the animal's pose better satisfies "its front and hind legs spread wide apart."

Our main contributions are:

• We introduce a lightweight external velocity-correction framework for fully frozen few-step text-to-image generators. To our knowledge, it is the first to learn velocity residuals with a corrector running in parallel with the generator while preserving its parameters, sampling schedule, and backbone forward-pass count.

• We empirically characterize how correction timing affects few-step generation. These findings motivate applying corrections at selected early steps, leaving subsequent denoising updates available to accommodate inaccuracies in the learned residuals.

• We develop On-Trajectory Flow-Bridge Supervision for state-dependent residual targets combining reference-velocity correction with trajectory recovery, and Shared Multi-Step Learning for training a single corrector across selected steps with local supervision.

• On our target complex-scene distribution, an 18.6M-parameter corrector achieves +17.60 pp NetWin over frozen Z-Image-Turbo, with only 0.84% additional inference latency and 0.04 GiB additional peak allocated GPU memory.

2 Related Work

Fine-Tuning for Text-to-Image Generation. Supervised and feedback-based fine-tuning improves text–image alignment: CoMat addresses concept omission and attribute binding, while CRAFT and DreamSync use curated or feedback-filtered supervision (Jiang et al., 2024; Sun et al., 2026; 2025). LoRA supports parameter-efficient adaptation (Hu et al., 2022). For few-step generators, fine-tuning can disrupt sampling behavior learned through specialized distillation objectives, including LCM, TCD, Hyper-SD, DMD2, and SDXL-Lightning (Luo et al., 2023; Zheng et al., 2024; Ren et al., 2024; Yin et al., 2024; Lin et al., 2024). Our method keeps the generator frozen and trains an auxiliary velocity corrector.

Reward- and Preference-Based Alignment. ImageReward, PickScore, and HPSv2 provide learned preference signals (Xu et al., 2023; Kirstain et al., 2023; Wu et al., 2023), while DDPO, AlignProp, and DRaFT optimize generators through reward feedback (Black et al., 2024; Prabhudesai et al., 2024; Clark et al., 2024). Diffusion-DPO, D3PO, and Diffusion-RPO support preference optimization without separately trained reward models (Wallace et al., 2024; Yang et al., 2024; Gu et al., 2024), with further developments in DiffusionNFT, TreeGRPO, and DiffusionOPSD (Zheng et al., 2026a; Ding & Ye, 2026; Zhou et al., 2026). Multiple compositional requirements and evaluator reliability complicate reward design (Chen et al., 2025; Li et al., 2026). Although generated and target images could also form preference pairs for Diffusion-DPO, we use target images to derive explicit stepwise velocity-residual supervision at corrected sampler states.

Trajectory-Based Adaptation and Control. ControlNet and T2I-Adapter add conditioning modules to frozen diffusion models (Zhang et al., 2023; Mou et al., 2024); Universal Guidance and FreeControl provide training-free control (Bansal et al., 2024; Mo et al., 2024); and D-Flow, FlowChef, and VESFlow intervene in flow trajectories or vector fields (Ben-Hamu et al., 2024; Patel et al., 2024; Choi & Yoon, 2026).

D-OPSD adapts step-distilled generators by matching target-image-informed teacher predictions on student rollouts (Jiang et al., 2026). We instead derive residual targets from the target latent and current frozen-generator prediction, without a teacher branch or generator updates.

Residual Learning corrects samples near the end of diffusion sampling (Zhang et al., 2024). Diffusion Controller conditions its side network on the reverse-transition mean from the current backbone prediction (Yang et al., 2026), introducing a within-step dependency. Our corrector uses the current latent and cached preceding-step velocity, enabling parallel execution with the current backbone evaluation.

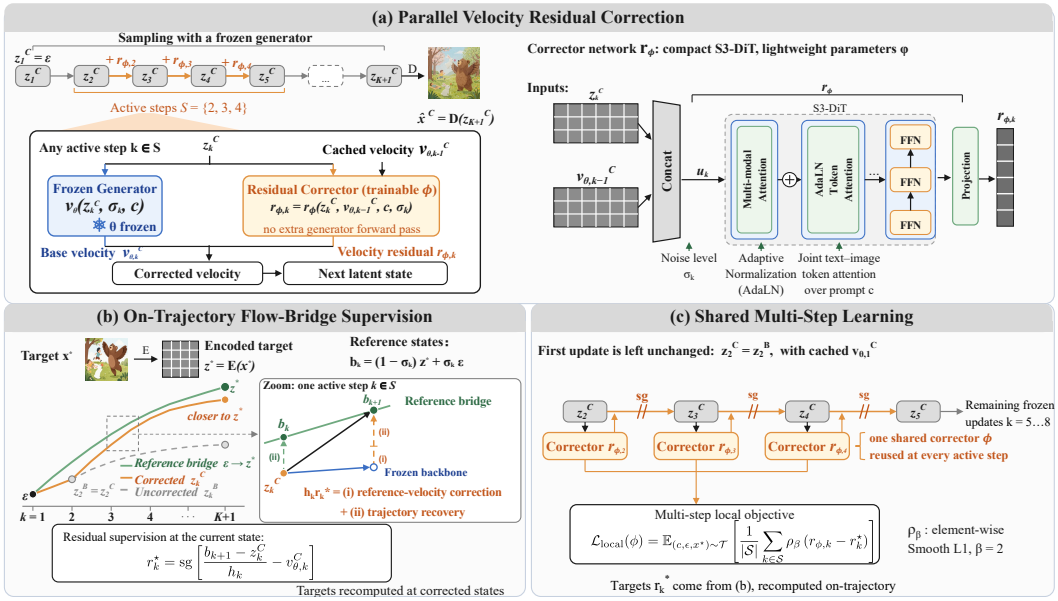


Figure 3: **Overview of velocity residual correction.** A lightweight corrector augments a frozen few-step generator without changing its sampling schedule or forward-pass count. **(a) Parallel Velocity Residual Correction.** Using the current latent and cached previous-step velocity, the corrector runs alongside the generator and adds a residual to its predicted velocity. **(b) On-Trajectory Flow-Bridge Supervision.** A noise-to-target reference bridge defines residual targets at corrected states, combining reference-velocity correction with trajectory recovery. **(c) Shared Multi-Step Learning.** One corrector is trained across selected steps with local supervision and no backpropagation through sampling updates. $r_{\phi,k}$ and $r_k^\star$ denote predicted and target residuals.

3 Method

As shown in Fig. 3, our framework adds velocity residuals at selected sampling steps of a frozen generator, preserving its sampling schedule and backbone forward-pass count.

Problem Setup. We consider few-step generators whose sampling updates admit an Euler-form velocity parameterization with a fixed noise schedule $\sigma_1 = 1 > \sigma_2 > \cdots > \sigma_K > \sigma_{K+1} = 0$, where K is the number of sampling updates. We instantiate the framework with Z-Image-Turbo using $K = 8$. Let z_k denote the latent state before update k, c the text condition, and $v_\theta(z_k, \sigma_k, c)$ the latent-space velocity predicted by the pretrained generator with frozen parameters θ. This velocity specifies how the latent state moves along the generation trajectory as the noise level changes.

The original sampling update is

$$z_{k+1}^B = z_k^B + h_k v_\theta(z_k^B, \sigma_k, c), \qquad h_k = \sigma_{k+1} - \sigma_k < 0, \tag{1}$$

where B denotes the uncorrected trajectory and $h_k < 0$ is the signed sampling step size as the noise level decreases. Starting from $z_1^B = \epsilon$, with $\epsilon \sim \mathcal{N}(0, I)$, the sampler produces the final latent z_{K+1}^B, which is decoded as $\hat{x}^B = \mathcal{D}(z_{K+1}^B)$ by the pretrained decoder $\mathcal{D}$. Throughout, $\mathcal{E}$ includes the backbone-specific latent normalization used by the pretrained generator, and $\mathcal{D}$ includes its inverse before VAE decoding. All states and bridge targets are expressed in this common coordinate system.

3.1 Parallel Velocity Residual Correction

Starting from $z_1^C = \epsilon$, we apply a corrector r_ϕ at steps $\mathcal{S} \subseteq \{2, \ldots, K\}$, leaving the first update unchanged. The frozen generator predicts $v_{\theta,k}^C = v_\theta(z_k^C, \sigma_k, c)$. At each active step, the corrector takes the current latent and cached preceding-step generator velocity as inputs:

$$r_{\phi,k} = r_\phi\left(z_k^C, v_{\theta,k-1}^C, c, \sigma_k\right), \qquad v_{\theta,k-1}^C = v_\theta(z_{k-1}^C, \sigma_{k-1}, c). \tag{2}$$

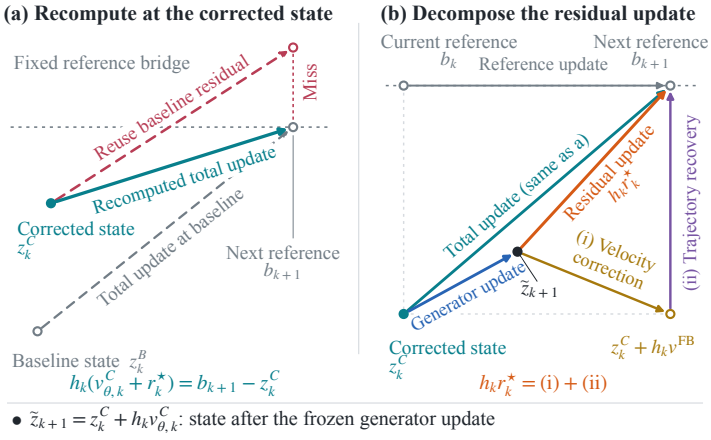


Figure 4: **On-Trajectory Flow-Bridge Supervision.** **(a)** Reusing a residual target from z_k^B at z_k^C would cause a mismatch (red dashed). We instead recompute targets at states visited by the corrected sampler during training. **(b)** The total update (teal in both panels) sums the generator update (blue) and residual update (orange), which decomposes into reference-velocity correction (gold) and trajectory recovery (purple). The black dot denotes the generator-only next state $\tilde{z}_{k+1} = z_k^C + h_k v_{\theta,k}^C$. Arrows denote state displacements.

We instantiate r_ϕ with a compact S3-DiT architecture following Z-Image Team et al. (2025). It takes the channel-wise concatenation $u_k = \mathrm{Concat}[z_k^C; v_{\theta,k-1}^C]$ as its latent input and incorporates σ_k through adaptive normalization. The corrector reuses the precomputed text token embeddings from the frozen generator's text encoder, requiring no additional text-encoding forward pass. These embeddings are jointly processed with the corrector's image tokens, enabling token-level text–image interactions.

The corrected velocity and sampling update are

$$\hat{v}_k = \begin{cases} v_{\theta,k}^C + r_{\phi,k}, & k \in \mathcal{S}, \\ v_{\theta,k}^C, & k \notin \mathcal{S}, \end{cases} \qquad z_{k+1}^C = z_k^C + h_k \hat{v}_k. \tag{3}$$

For a fixed generator, the current velocity is a deterministic function of (z_k^C, σ_k, c), all available to the corrector; the cached previous-step velocity supplies an additional trajectory-dependent feature. The corrector thus learns the residual without taking the current generator output as an input, allowing both branches to execute concurrently. The current output is used to construct the training target in Sec. 3.2 and combined with the predicted residual in Eq. (3).

3.2 On-Trajectory Flow-Bridge Supervision

During training, corrected states z_k^C are generated online by rolling out the sampler with the current corrector, rather than being precomputed from the baseline trajectory. Before the first active correction, $z_2^C = z_2^B$; later corrected states are produced by preceding corrected updates. Therefore, a residual target computed at the baseline state z_k^B may no longer be valid at the actually visited state z_k^C. As illustrated in Fig. 4(a), we instead recompute the required velocity correction at each corrected state using a fixed reference path from the initial noise to the target latent.

Given the target image $x^\star$, we obtain its clean latent $z^\star = \mathcal{E}(x^\star)$ using the pretrained encoder $\mathcal{E}$. With the initial noise ϵ used to generate the original image, we define

$$b_k = (1 - \sigma_k) z^\star + \sigma_k \epsilon. \tag{4}$$

This linear reference path connects $b_1 = \epsilon$ to $b_{K+1} = z^\star$ and need not follow the generator's trajectory. It remains fixed for each training example, while residual targets are recomputed at the changing corrected states.

At each active step, the online rollout provides the current corrected state z_k^C, at which we evaluate the frozen generator to obtain $v_{\theta,k}^C$. The velocity required to reach b_{k+1} in one Euler update is

$(b_{k+1} - z_k^C)/h_k$. Subtracting the generator prediction gives the residual target

$$r_k^\star = \mathrm{sg}\left[\frac{b_{k+1} - z_k^C}{h_k} - v_{\theta,k}^C\right], \tag{5}$$

where $\mathrm{sg}[\cdot]$ denotes stop-gradient.

To explain how this target accounts for both the generator prediction and the current state, we decompose it as

$$r_k^\star = \mathrm{sg}\left[\underbrace{v^{\mathrm{FB}} - v_{\theta,k}^C}_{\text{reference-velocity correction}} + \underbrace{\frac{b_k - z_k^C}{h_k}}_{\text{trajectory recovery}}\right], \qquad v^{\mathrm{FB}} = \frac{b_{k+1} - b_k}{h_k} = \epsilon - z^\star. \tag{6}$$

The first term corrects the discrepancy between the reference velocity and the generator prediction, while the second compensates for displacement of the current state from the reference bridge. Figure 4(b) illustrates these terms as state displacements: multiplying them by h_k gives the reference-velocity correction and trajectory-recovery arrows, whose sum is the required residual update $h_k r_k^\star$.

For a fixed current state, the residual prediction error directly determines the next-state bridge error:

$$z_{k+1}^C - b_{k+1} = h_k\left(r_{\phi,k} - r_k^\star\right), \qquad k \in \mathcal{S}. \tag{7}$$

An exact residual prediction therefore reaches b_{k+1} in one step, but subsequent uncorrected updates need not remain on the bridge. During training, we roll out the corrected sampler and recompute the residual target in Eq. (5) at every active step.

3.3 Shared Multi-Step Learning

We share a single noise-conditioned corrector across selected steps and train it with local supervision. Eq. (7) shows that each visited state provides a supervised training instance without differentiating through the preceding sampling updates.

We zero-initialize the corrector's output projection so that training starts from the original generator trajectory. Our default active-step set is $\mathcal{S} = \{2, 3, 4\}$, with the previous-step generator velocity cached as input to the corrector.

For paired training tuples $(c, \epsilon, x^\star) \sim \mathcal{T}$, we minimize

$$\mathcal{L}_{\mathrm{local}}(\phi) = \mathbb{E}_{(c,\epsilon,x^\star)\sim\mathcal{T}}\left[\frac{1}{|\mathcal{S}|}\sum_{k\in\mathcal{S}}\rho_\beta\left(r_{\phi,k} - r_k^\star\right)\right], \tag{8}$$

where $\mathcal{T}$ denotes the training set and ρ_β is the Smooth L1 loss averaged over latent elements. We use $\beta = 2$ by default and weight active steps equally in velocity-residual space.

We detach rollout states and cached velocities between steps and treat the generator predictions and residual targets as constants. Thus, later losses are evaluated on states produced by earlier corrections without backpropagating through the preceding corrections or sampling updates.

The pretrained generator, text encoder, and latent encoder–decoder remain frozen throughout training. At inference, the shared corrector is applied at the same active steps, and the generator completes the remaining updates before decoding $\hat{x}^C = \mathcal{D}(z_{K+1}^C)$.

4 Experiments

4.1 Experimental Setup

Data. We use approximately 4K compositional captions with a 90/10 training/validation split. Offline target images are generated by editing frozen-backbone outputs with GPT-Image-2 (OpenAI, 2026). Each target is paired with the original caption and the initial noise used to generate the backbone output. Evaluation uses 574 held-out prompts with no normalized exact overlap with either the training or validation prompts (Appendix A.1).

Table 1: Results on Z-Image-Turbo. NetWin is measured against the frozen baseline on 574 held-out target-domain prompts. All metrics are higher-is-better; best scores are bold, and second-best scores achieved by our method are underlined.

		Target-domain	General-purpose							
Method	Trainable	NetWin (pp)	Color	Shape	Texture	2D Spatial	3D Spatial	Numeracy	Non-spatial	Complex
Frozen baseline	–	0.00	0.7919	0.5731	0.7350	0.3349	**0.4158**	0.6738	0.3121	0.3927
SFT + LoRA	19.85M	3.83	0.7962	0.5706	0.7351	0.3359	0.4156	0.6704	0.3125	0.3947
DPO + LoRA	19.85M	1.05	0.7929	0.5722	0.7347	**0.3375**	0.4148	0.6757	0.3122	0.3928
D-OPSD	19.85M	-12.37	0.7597	0.5631	0.7206	0.3286	0.4147	0.6382	0.3109	0.3880
DiffCon adaptation	20.21M	1.74	0.7827	0.5843	**0.7495**	0.3305	0.4133	**0.6838**	0.3119	0.3938
Ours	18.6M	**17.60**	**0.8171**	**0.5887**	0.7440	0.3361	0.4107	0.6748	**0.3132**	**0.4026**



Figure 5: Compositional errors in few-step generation and qualitative comparisons. Our method better satisfies the highlighted requirements than competing methods.

Implementation and baselines. We equip frozen eight-step Z-Image-Turbo (Z-Image Team et al., 2025) with an 18.6M-parameter S3-DiT corrector shared across $\mathcal{S} = \{2, 3, 4\}$. Only the corrector is trained, using AdamW with learning rate 2×10^{-4} and Smooth L1 loss ($\beta = 2$), without backpropagation through rollout updates; the backbone, text encoder, and VAE remain frozen. We compare against frozen Turbo, SFT+LoRA (Hu et al., 2022), DPO+LoRA (Wallace et al., 2024), D-OPSD (Jiang et al., 2026), and DiffCon (Yang et al., 2026), using the same backbone and velocity parameterization and evaluating all methods with eight sampling steps. SFT+LoRA, DPO+LoRA, and D-OPSD each use 19.85M trainable parameters, while DiffCon uses a 20.21M-parameter controller. Architecture, optimization, and baseline implementation details are provided in Appendix A.2.

Evaluation metrics. We conduct Good/Same/Bad (GSB) comparisons against frozen Turbo on 574 held-out target-domain prompts with matched seeds and both image orders. We report $\mathrm{NetWin} = 100(G - B)/N$ in percentage points (pp), where G, S, and B denote adapted-model wins, ties, and baseline wins, respectively, and $N = G + S + B = 574$. GSB is used only for evaluation; judging details and repeatability results appear in Appendix A.3. We additionally report category-wise scores on T2I-CompBench++ (Huang et al., 2025), using the complete official test set with four images per prompt. Evaluator and sampling details appear in Appendix A.4.

Inference efficiency. We measure end-to-end latency on one NVIDIA A800 at batch size one, BF16 precision, and 1024×1024 resolution. Each method uses the same 50 held-out prompts and three seeds (150 generations), with eight backbone evaluations per image. Timing includes text encoding, denoising, and VAE decoding, but excludes model loading and image saving. We report absolute and relative increases in mean latency across generations, together with the increase in peak allocated GPU memory, all relative to frozen Z-Image-Turbo.

4.2 Main Results

Our 18.6M-parameter corrector achieves the highest target-domain NetWin (+17.60 pp) among the evaluated methods (Table 1). On T2I-CompBench++, it leads in four categories and ranks second in two, improving seven of eight scores over frozen Z-Image-Turbo with a small decrease in 3D spatial relations. Figure 5 illustrates improved adherence to compositional requirements relative to the evaluated baselines.

Under the unchanged eight-step schedule, our method adds only 0.84% end-to-end latency (26.9 ms/image) and 0.04 GiB peak allocated GPU memory, achieving higher target-domain NetWin with lower latency overhead than the auxiliary-network baseline DiffCon (Table 2). The example in Fig-

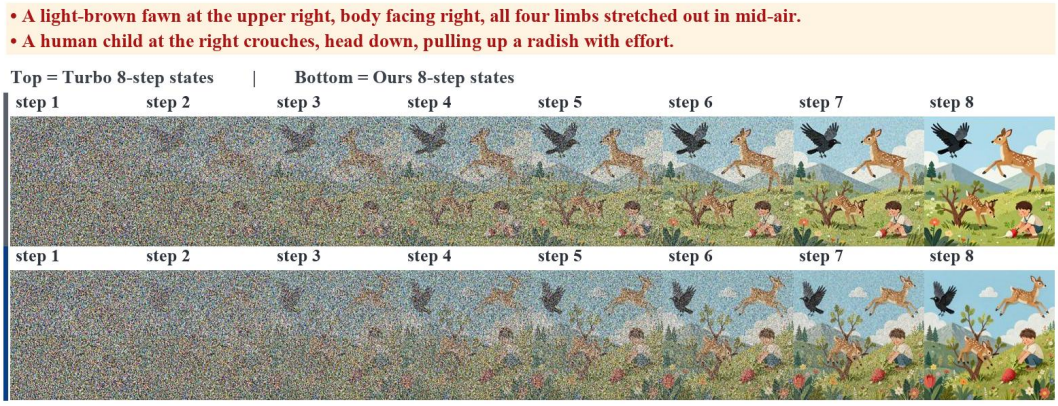


Figure 6: Step-wise qualitative comparison with frozen Z-Image-Turbo. The upper row shows the baseline and the lower row shows our method across eight sampling steps.

Table 2: Target-domain NetWin and inference overhead relative to frozen Z-Image-Turbo. Bold values indicate the best results among adaptation methods.

Method	NetWin (pp)↑	Latency overhead (%)↓	Added latency (ms)↓	Memory increase (GiB)↓
Frozen Turbo	0.00	0.00	0.0	0.00
DiffCon	+1.74	3.24	103.9	**0.02**
Ours	**+17.60**	**0.84**	**26.9**	0.04

ure 6 shows improved pose and action details emerging during intermediate sampling and persisting in the final image.

4.3 Ablation Studies

Ablation setup. All ablations use Z-Image-Turbo and the same training subset, reporting GSB proportions and NetWin against the frozen baseline on 574 held-out prompts with matched seeds. Unless otherwise stated, we use corrected-rollout Flow-Bridge supervision, previous-velocity conditioning, and one corrector shared across $\mathcal{S} = \{2, 3, 4\}$, trained with Smooth L1 loss without backpropagation through rollout updates.

State–target pairing. We test whether effective correction requires both corrected-rollout states and targets recomputed at those states (Figure 7(a)). With targets computed at each respective state, training on GT-bridge states yields little improvement, base-rollout training degrades performance, and corrected-rollout training produces the strongest gains. To separate rollout choice from target recomputation, we compare base and corrected rollouts using the same fixed target $r_k^{\text{fixed}} = \text{sg}[v^{\text{FB}} - v_\theta(b_k, \sigma_k, c)]$, obtained by evaluating Eq. (5) at b_k; cached velocities come from the respective rollouts. With this fixed target, corrected rollouts largely remove degradation but yield little improvement over the baseline. Recomputing targets at corrected states then produces substantial gains, supporting the joint use of corrector-induced states and state-dependent supervision.

Flow-Bridge target decomposition. Following Eq. (6), we compare reference-velocity correction $a_k = v^{\text{FB}} - v^C_{\theta,k}$, trajectory recovery $q_k = (b_k - z^C_k)/h_k$, and their sum under identical corrected-rollout training, with all targets detached. Both components improve NetWin individually, while the full target outperforms either alone (Figure 7(b)), supporting their complementary roles. Combining them increases both win and loss rates, with the larger rise in wins yielding the highest NetWin.

Velocity conditioning. We compare zeroed velocity channels, current velocity $v^C_{\theta,k}$, and cached previous velocity $v^C_{\theta,k-1}$, holding the remaining configuration fixed (Figure 7(c)). Both velocity-conditioned variants outperform zeroed conditioning. Previous velocity achieves the highest observed NetWin, with a similar win rate and a lower loss rate than current velocity. It also removes dependence on the current backbone output, enabling parallel execution.

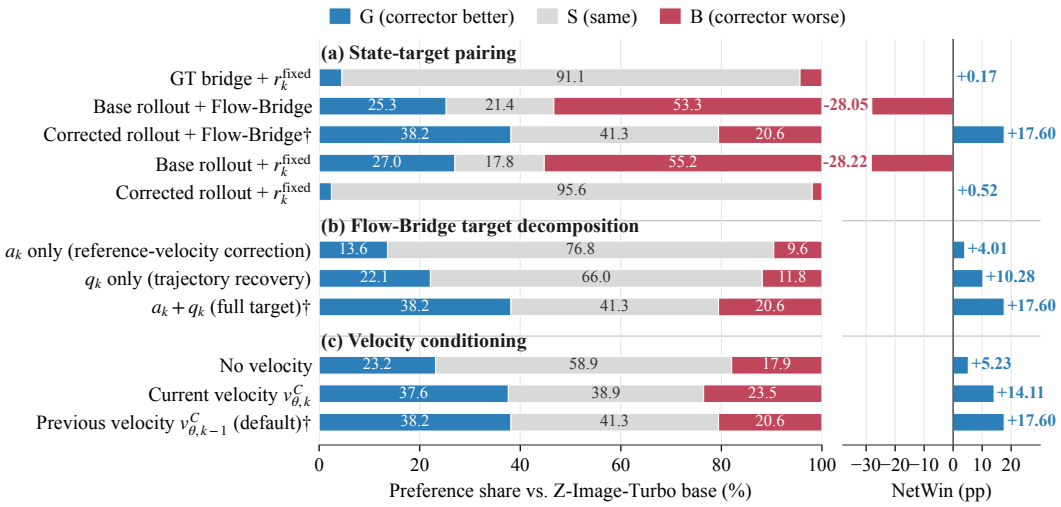
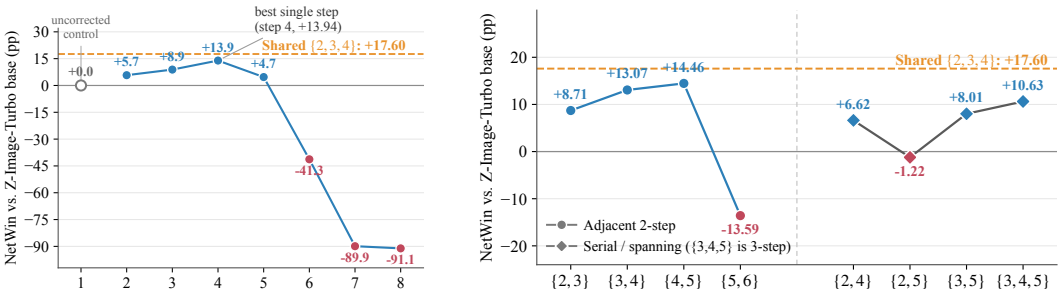


Figure 7: Ablations on Z-Image-Turbo: (a) state–target pairing, (b) Flow-Bridge target decomposition, and (c) velocity conditioning. We report GSB proportions (%, left) and NetWin (pp, right) against frozen Z-Image-Turbo. † marks the full-method result repeated for reference.



(a) Single-step correction.

(b) Multi-step correction.

Figure 8: Active-step ablations under eight-step Z-Image-Turbo sampling: (a) single-step and (b) multi-step correction. NetWin is evaluated against the frozen baseline on 574 held-out prompts with matched seeds. The orange dashed line denotes the default $\mathcal{S} = \{2, 3, 4\}$ configuration (+17.60 pp).

Active-step configurations. Under the same eight-step schedule, the default $\mathcal{S} = \{2, 3, 4\}$ achieves the highest NetWin among tested configurations, exceeding the best single-step ($\{4\}$) and two-step ($\{4, 5\}$) variants (Figure 8). More correction steps do not necessarily help: $\{3, 4, 5\}$ underperforms $\{4, 5\}$, and late-step correction can severely degrade performance. This may reflect fewer remaining updates to accommodate correction-induced deviations, supporting intervention at selected early steps.

5 Conclusion

We introduced a lightweight parallel velocity corrector trained with on-trajectory Flow-Bridge supervision, preserving the frozen generator's parameters, sampling schedule, and generator forward-pass count. With fewer than 20M trainable parameters, it achieves +17.60 pp target-domain NetWin on Z-Image-Turbo with minimal inference overhead.

Very short schedules, such as four-step generation, may leave insufficient updates after correction to complete denoising. Offline targets from the proprietary GPT-Image-2 editor may also introduce biases and limit reproducibility. Future work will explore schedule-aware correction across few-step generators and reproducible supervision from open models.

AI USE STATEMENT

We used generative AI tools for data construction, automatic evaluation, and manuscript preparation. For data construction, `claude-opus-4-8` assisted in expanding structured scene specifications into prompts and checking their consistency, atomic requirements, and subject assignments. GPT-Image-2 was used to edit frozen-generator outputs to construct offline target images for training. For GSB evaluation, `gpt-5-chat-0807` extracted visual requirements, and `claude-opus-4-8` assessed requirement satisfaction and severe structural failures. These uses are described in the data construction and evaluation sections and their corresponding appendices. We also used LLMs to assist with translation, language polishing, and refinement of the manuscript's organization and presentation.

The authors reviewed the AI-assisted manuscript and checked that its technical descriptions and reported results were consistent with the implemented methods and experimental records. AI-generated data and automatic judgments were processed using the checking and evaluation procedures described in the paper. The authors take responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

REFERENCES

Anthropic. Introducing Claude Opus 4.8, May 2026. URL `https://www.anthropic.com/news/claude-opus-4-8`. Published May 28, 2026. Accessed September 16, 2026.

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Roni Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *International Conference on Learning Representations*, volume 2024, pp. 51304–51323, 2024.

Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-Flow: Differentiating through flows for controlled generation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 3462–3483. PMLR, 2024.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, 2024.

Zhaorun Chen, Zichen Wen, Yichao Du, Yiyang Zhou, Chenhang Cui, Siwei Han, Jen Weng, Chaoqi Wang, Zhengwei Tong, Leria Huang, Canyu Chen, Haoqin Tu, Qinghao Ye, Zhihong Zhu, Yuqing Zhang, Jiawei Zhou, Zhuokai Zhao, Rafael Rafailov, Chelsea Finn, and Huaxiu Yao. MJ-Bench: Is your multimodal reward model really a good judge for text-to-image generation? In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Yujin Choi and Jaehong Yoon. Safe few-step generation via velocity editing. *arXiv preprint arXiv:2606.23267*, 2026.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *International Conference on Learning Representations*, 2024.

DiffSynth-Studio Contributors. DiffSynth-Studio. `https://github.com/modelscope/DiffSynth-Studio`, 2023. Open-source software.

Zheng Ding and Weirui Ye. TreeGRPO: Tree-advantage GRPO for online RL post-training of diffusion models. In *International Conference on Learning Representations*, 2026.

Yi Gu, Zhendong Wang, Yueqin Yin, Yujia Xie, and Mingyuan Zhou. Diffusion-RPO: Aligning diffusion models through relative preference optimization. *arXiv preprint arXiv:2406.06382*, 2024.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7514–7528. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.595.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Kaiyi Huang, Chengqi Duan, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2I-CompBench++: An enhanced and comprehensive benchmark for compositional text-to-image generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. doi: 10.1109/TPAMI.2025.3531907.

Dengyang Jiang, Dongyang Liu, Zanyi Wang, Qilong Wu, Xin Jin, David Liu, Zhen Li, Mengmeng Wang, Peng Gao, and Harry Yang. Distribution matching distillation meets reinforcement learning. *arXiv preprint arXiv:2511.13649*, 2025.

Dengyang Jiang, Xin Jin, Dongyang Liu, Zanyi Wang, Mingzhe Zheng, Ruoyi Du, Xiangpeng Yang, Qilong Wu, Zhen Li, Peng Gao, Harry Yang, and Steven Hoi. D-OPSD: On-policy self-distillation for continuously tuning step-distilled diffusion models. *arXiv preprint arXiv:2605.05204*, 2026.

Dongzhi Jiang, Guanglu Song, Xiaoshi Wu, Renrui Zhang, Dazhong Shen, Zhuofan Zong, Yu Liu, and Hongsheng Li. CoMat: Aligning text-to-image diffusion model with image-to-text concept matching. In *Advances in Neural Information Processing Systems*, volume 37, pp. 76177–76209, 2024.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *International Conference on Learning Representations*, 2024.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-Pic: An open dataset of user preferences for text-to-image generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 12888–12900. PMLR, 2022. URL https://proceedings.mlr.press/v162/li22n.html.

Niantong Li, Guangzheng Hu, Weixu Qiao, Ying Ba, Qichen Hong, Shijun Shen, Jinlin Wang, Fan Zhou, Jianye Kang, Xin Shang, Ziyi He, Wei Wang, Dalin Li, Jiahao Li, Jie Zhang, Kaiyuan Gao, Kun Yan, Lihan Jiang, Ningyuan Tang, Shengming Yin, Tianhe Wu, Xiao Xu, Xiaoyue Chen, Yuxiang Chen, Yan Shu, Yanran Zhang, Yilei Chen, Yixian Xu, Zekai Zhang, Zhendong Wang, Zihao Liu, Zikai Zhou, Hongzhu Shi, Yi Wang, Bing Zhao, Hu Wei, Lin Qu, and Chenfei Wu. Qwen-Image-Bench: From generation to creation in text-to-image evaluation. *arXiv preprint arXiv:2605.28091*, 2026.

Shanchuan Lin, Anran Wang, and Xiao Yang. SDXL-Lightning: Progressive adversarial diffusion distillation. *arXiv preprint arXiv:2402.13929*, 2024.

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.

Sicheng Mo, Fangzhou Mu, Kuan Heng Lin, Yanli Liu, Bochen Guan, Yin Li, and Bolei Zhou. FreeControl: Training-free spatial control of any text-to-image diffusion model with any condition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7465–7475, 2024.

Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. T2I-Adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 4296–4304, 2024. doi: 10.1609/aaai.v38i5.28226.

OpenAI. GPT-5 Chat Model. OpenAI API documentation. URL https://developers.openai.com/api/docs/models/gpt-5-chat-latest. Accessed September 16, 2026.

OpenAI. GPT-Image-2 model. https://developers.openai.com/api/docs/models/gpt-image-2, 2026. Accessed September 14, 2026.

Maitreya Patel, Song Wen, Dimitris N. Metaxas, and Yezhou Yang. Steering rectified flow models in the vector field for controlled image generation. *arXiv preprint arXiv:2412.00100*, 2024.

Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation. *arXiv preprint arXiv:2310.03739v2*, 2024.

Yuxi Ren, Xin Xia, Yanzuo Lu, Jiacheng Zhang, Jie Wu, Pan Xie, Xing Wang, and Xuefeng Xiao. Hyper-SD: Trajectory segmented consistency model for efficient image synthesis. In *Advances in Neural Information Processing Systems*, volume 37, pp. 117340–117362, 2024.

Jiao Sun, Deqing Fu, Yushi Hu, Su Wang, Royi Rassin, Da-Cheng Juan, Dana Alon, Charles Herrmann, Sjoerd van Steenkiste, Ranjay Krishna, and Cyrus Rashtchian. DreamSync: Aligning text-to-image generation with image understanding feedback. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5920–5945, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics.

Zening Sun, Zhengpeng Xie, Lichen Bai, Shitong Shao, Shuo Yang, and Zeke Xie. CRAFT: Aligning diffusion models with fine-tuning is easier than you think. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8228–8238, 2024.

Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. ImageReward: Learning and evaluating human preferences for text-to-image generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 15903–15935, 2023.

Kai Yang, Jian Tao, Jiafei Lyu, Chunjiang Ge, Jiaxin Chen, Weihan Shen, Xiaolong Zhu, and Xiu Li. Using human feedback to fine-tune diffusion models without any reward model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8941–8951, 2024.

Tong Yang, Moonkyung Ryu, Chih-Wei Hsu, Guy Tennenholtz, Yuejie Chi, Craig Boutilier, and Bo Dai. Diffusion controller: Framework, algorithms and parameterization. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Frédo Durand, and William T. Freeman. Improved distribution matching distillation for fast image synthesis. In *Advances in Neural Information Processing Systems*, volume 37, pp. 47455–47487, 2024.

Z-Image Team, Huanqia Cai, Sihan Cao, Ruoyi Du, Peng Gao, Aiming Hao, Steven Hoi, Zhaohui Hou, Shijie Huang, Dengyang Jiang, Yuming Jiang, Xin Jin, Liangchen Li, Zhen Li, Zhong-Yu Li, David Liu, Dongyang Liu, Qilong Wu, Feng Yu, Zechao Zhan, Chi Zhang, Shifeng Zhang, Ruikai Zhou, and Shilin Zhou. Z-Image: An efficient image generation foundation model with single-stream diffusion transformer. *arXiv preprint arXiv:2511.22699*, 2025.

Junyu Zhang, Daochang Liu, Eunbyung Park, Shichao Zhang, and Chang Xu. Residual learning in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7289–7299, 2024.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3836–3847, 2023.

Jianbin Zheng, Minghui Hu, Zhongyi Fan, Chaoyue Wang, Changxing Ding, Dacheng Tao, and Tat-Jen Cham. Trajectory consistency distillation: Improved latent consistency distillation by semi-linear consistency function with trajectory mapping. *arXiv preprint arXiv:2402.19159*, 2024.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. DiffusionNFT: Online diffusion reinforcement with forward process. In *International Conference on Learning Representations*, 2026a.

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. In *International Conference on Learning Representations*, 2026b.

Wei Zhou, Xiongwei Zhu, Lingdong Kong, Bo Chen, Lei Zhang, Yongyuan Liang, Xiaoxia Hou, Ye Tian, Xian Sun, Yingshuo Wang, Linfeng Li, Shengqiong Wu, Leigang Qu, Feng Li, Wei Liu, Julian McAuley, and Tat-Seng Chua. On-policy self-distillation in diffusion models. *arXiv preprint arXiv:2608.24646*, 2026.

Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Simple multi-dataset detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. URL `https://github.com/xingyizhou/UniDet`.

A Additional Experimental Details

A.1 Data Construction and Overlap Checks

We construct approximately 4K compositional captions by sampling structured scene specifications from curated pools of subjects, scenes, and actions under predefined constraints, and using `claude-opus-4-8` (Anthropic, 2026) to expand them into descriptions. The descriptions undergo consistency checks, verification of decomposed atomic requirements, and checks for ambiguous subject assignment, with revisions as needed, before being formatted into final image-generation prompts. The resulting captions cover multi-subject scenes, human actions, interactions, spatial relations, counting, and attribute binding.

For each caption, we sample an initial noise ϵ and generate an image with the frozen backbone. The generated image and caption are provided to the GPT-Image-2-based editing model `gpt-image-2-0421-global-edit` (OpenAI, 2026). The edited image serves as the supervision target, while the original caption and initial noise are retained to form the training tuple $(c, \epsilon, x^\star)$. The editor is used only for offline target construction and is not queried during corrector training or inference.

We reserve 10% of the caption pool for validation and use the remainder for training. Final target-domain evaluation uses 574 held-out prompts. Exact string matching after whitespace normalization confirms that these evaluation prompts are unique and do not overlap with either the training or validation set.

A.2 Architecture, Optimization, and Baselines

Corrector architecture. The S3-DiT corrector contains six Transformer blocks with hidden dimension 384, three attention heads, head dimension 128, and patch size two, totaling approximately 18.6M trainable parameters. A single corrector is shared across $\mathcal{S} = \{2, 3, 4\}$ of the frozen eight-step Z-Image-Turbo generator. It reuses the generator's precomputed text token embeddings without an additional text-encoding forward pass, and the generator and corrector branches execute in parallel at active steps.

Corrector optimization. We train only the corrector using AdamW with learning rate 2×10^{-4}, weight decay 10^{-2}, effective batch size 64, and gradient-norm clipping at 1.0. Training uses BF16 precision, a per-GPU micro-batch size of one, gradient checkpointing, and the Smooth L1 residual objective with $\beta = 2$. The backbone, text encoder, and VAE remain frozen, and rollout states and cached velocities are detached between successive sampling steps, as described in Section 3.3.

Baseline configuration. Our implementation uses DiffSynth-Studio (DiffSynth-Studio Contributors, 2023). The three LoRA baselines use rank $r = 8$, scaling factor $\alpha = 8$, and zero dropout, with adapters applied to the attention query, key, value, and output projections and the three feed-forward projections across the 30 main, two noise-refiner, and two context-refiner blocks. Each has 19,845,120 trainable student parameters; the original backbone weights, text encoder, and VAE remain frozen. All four trainable baselines use AdamW with a configured learning rate of 10^{-5}, weight decay 10^{-2}, and effective batch size 64, measured in preference pairs for DPO+LoRA.

SFT+LoRA. We train the adapters by flow matching on paired image–text data. Given a target latent x_0 and Gaussian noise ϵ, we construct $x_\sigma = (1 - \sigma)x_0 + \sigma\epsilon$ and regress the velocity target $\epsilon - x_0$. Training samples an index uniformly from a 1,000-point noise grid with shift three and uses the implementation's normalized Gaussian-envelope loss weighting.

DPO+LoRA. We perform offline diffusion preference optimization with target images as chosen samples and fixed eight-step frozen-Turbo generations from the same prompts as rejected samples. Chosen and rejected samples share the same noise and noise level, and the frozen reference predictions are obtained by disabling the LoRA adapters. We use preference coefficient $\beta_{\mathrm{DPO}} = 2000$ and select the checkpoint with the lowest validation DPO loss on a prompt-disjoint holdout.



Figure 9: Additional qualitative comparisons of compositional instruction following in few-step text-to-image generation, supplementing Fig. 5.

D-OPSD. We train student LoRA adapters along eight-step student-generated trajectories by matching the denoised latent estimates of an EMA teacher, with trajectory states detached between steps. The teacher is conditioned on the prompt and target image through a frozen Qwen3-VL-4B-Instruct model, and its adapters are updated with EMA decay 0.9999. Teacher parameters are excluded from the trainable parameter count. Inference uses only the student and the original text encoder.

DiffCon. We train an independent controller with eighteen Transformer blocks, hidden dimension 256, two attention heads, and 20,206,401 trainable parameters (20.21M). It receives the current noisy latent and the frozen backbone's denoised estimate, together with text and timestep conditioning, and predicts a scalar z and residual field h to form $v_{\text{DiffCon}} = v_{\text{base}} + (1 - z)h - zv_{\text{base}}$. Its output projection is zero-initialized. Training uses flow-matching mean squared error on noised target latents, sampling uniformly from the eight inference noise levels. At inference, the controller is evaluated after the backbone at every sampling step.

A.3 GSB Judging Details and Repeatability

Evaluation consists of two stages. First, `gpt-5-chat-0807` (OpenAI) decomposes each prompt into independently verifiable visual requirements, annotated by importance (core or detail) and dimension. Second, the visual judge `claude-opus-4-8` (Anthropic, 2026) compares each image pair requirement by requirement and checks for severe structural failures. Images are resized to a longest side of 768 pixels before judging.

Each pair is evaluated in both image orders. After mapping judgments back to model identities, we retain a directional requirement preference if at least one order supports it and neither order supports the opposite direction. Sample-level outcomes are determined lexicographically by severe-failure counts, net core-requirement preferences, and net detail-requirement preferences. Atmosphere requirements are recorded but excluded from the decision.

For each judging run, NetWin is computed using $\text{NetWin} = 100(G - B)/N$. Ties remain in the denominator, so the metric reflects both the direction of preferences and the prevalence of pairs without a decisive difference.

To assess automatic-judging stability, we perform five independent re-evaluations of the same fixed image pairs, applying the image-order reversal and aggregation protocol in each run. The stan-

dard deviation of the resulting run-level NetWin scores is below 1 percentage point. This measures judging repeatability on fixed outputs. GSB judgments are used only for evaluation and never as a corrector-training signal.

A.4 T2I-CompBench++ Evaluation Details

We evaluate compositional generalization on T2I-CompBench++ (Huang et al., 2025) using its official automatic evaluators: disentangled BLIP-VQA based on BLIP (Li et al., 2022) for color, shape, and texture binding; UniDet-based metrics (Zhou et al., 2022) for 2D/3D spatial relationships and numeracy; CLIPScore (Hessel et al., 2021) for non-spatial relationships; and 3-in-1 for complex compositions.

We use the complete official test prompt set, comprising 300 prompts in each of eight categories. Our evaluation differs from the official protocol only in the number of generated images per prompt: we generate four images (seeds 0–3) instead of ten. We report the mean score over all 1,200 generated images in each category.

A.5 Additional Qualitative Comparisons

Figure 9 supplements Fig. 5 with additional qualitative comparisons of compositional instruction following.