# WaDi: Weight Direction-aware Distillation for One-step Image Synthesis (*Supplementary Material*)

Lei Wang[1], Yang Cheng[1], Senmao Li[1], Ge Wu[1], Yaxing Wang[1,3†], Jian Yang[1,2†]
[1]PCA Lab, VCIP, College of Computer Science, Nankai University
[2] PCA Lab, School of Intelligence Science and Technology, Nanjing University
[3] NKIARI, Shenzhen Futian
{scitop1998, cyrene0613, senmaonk, gewu.nku}@gmail.com, {yaxing,csjyang}@nankai.edu.cn
Code: https://github.com/gudaochangsheng/WaDi

Table A1. Comparison of different methods in terms of memory, number of trainable parameters, FID, CLIP score, and latency.

| Type | Memory (M) | #Train Param. | FID | CLIP | Latency |
|---|---|---|---|---|---|
| LoRA | 1259 | 120.9M | 25.27 | 0.29 | 0.11s |
| LoRaD | 2021 | **83.8M** | **20.86** | **0.31** | 0.11s |
| FT (DMD2) | 17397 | 860M | 23.30 | 0.30 | 0.11s |

## A. Limitations and Future Work

Assuming the weight matrix $W \in \mathbb{R}^{d\times k}$, the training memory usage of full fine-tuning (FT) is $2 \times d \times k$, while that of LoRA is $2 \times (d \times r + r \times k)$. In comparison, LoRaD requires $d\times k+2\times\left(\frac{d}{2} \times r + r \times k\right) = d\times k+d\times r+2\times r\times k$, reflecting higher memory overhead than LoRA due to the need to reconstruct a full-rank rotated weight matrix during forward pass. Note that this operation is performed only in the forward pass and does not incur additional gradient storage. As shown in Tab. A1, LoRA uses $\sim$7% of DMD2's memory, while LoRaD uses slightly more at $\sim$12%. Future work may explore more memory-efficient rotation strategies that avoid explicit construction of the full rotated matrix. Additionally, given the flexibility and generality LoRaD has demonstrated in the image customization task, we plan to further investigate its applicability to a broader range of model fine-tuning tasks.

## B. Statement

**LLM Usage Statement.** We only employed a large language model (ChatGPT) for polishing the language and improving the clarity of expression. The model was not involved in the conception or design of the research, the development or execution of methods or experiments, data processing or statistical analysis, nor in the interpretation of results or the derivation of conclusions; therefore, it had no substantive impact on the core scientific contributions of this work.

**Ethics Statement.** We fully recognize the potential ethical risks associated with deploying generative models, including privacy breaches, data misuse, and the amplification of biases. At the same time, we acknowledge the potential misuse of personalization and customization techniques in generating false content and disinformation. To address these concerns, we advocate for and support responsible research and application practices, strictly adhering to relevant laws, regulations, and industry guidelines, while implementing necessary technical and governance measures to minimize the risks of misuse.

**Reproducibility Statement.** To promote reproducibility, we will release all source code and scripts after the peer review process, enabling others to replicate the experiments. All experiments in this work were conducted using publicly available datasets.

## C. Broader Impacts

WaDi compresses the inference process of diffusion models into one-step through knowledge distillation, significantly reducing computational resource consumption. At the same time, it promotes creativity among content generators and lowers the barrier to entry. However, the technology may also be misused to generate false or harmful images, thereby spreading misinformation and raising concerns related to copyright and intellectual property.

## D. More Theoretical Explanation

While our empirical results demonstrate that knowledge distillation primarily preserves weight directions, we further provide a theoretical perspective to support this observation. When optimizing only the directional component of weights, the parameters are constrained to lie on a unit hypersphere—a compact and smooth Riemannian manifold [1, 5].

†Corresponding authors.

According to manifold optimization theory [9, 17], such geometric constraints improve gradient flow and mitigate sharp or pathological minima, thereby stabilizing training.

In addition, constraining optimization to the hypersphere reduces the degrees of freedom in parameter space, serving as an implicit regularizer. This aligns with modern generalization theory in overparameterized settings, which suggests that limiting parameter complexity improves robustness [3, 4]. LoRaD explicitly models directional changes through learnable 2D rotations, analogous to reparameterizations used in weight normalization [30], which have been shown to accelerate convergence and enhance generalization [18, 30].

Optimizing directions alone substantially reduces sensitivity to weight norm and tends to produce smoother loss landscapes [23], favoring flatter minima—empirically associated with better generalization [2, 13]. These effects are particularly important for robustness under distribution shifts [8]. This theoretical grounding is consistent with our empirical findings, including the superior performance of LoRaD shown in Tab. 1 and its convergence behavior in Fig. A7 and Fig. A8.

## E. More Details on Weight Analysis

In our weight analysis, we decompose the weight matrix $W \in \mathbb{R}^{d\times k}$ into a norm vector and direction matrix as follows:

$$W = \eta\mathcal{V} \tag{1}$$

where $\eta \in \mathbb{R}^{1\times k}$ represents the column-wise norm, and $\mathcal{V} \in \mathbb{R}^{d\times k}$ denotes the normalized direction matrix.

To quantify the difference between the multi-step and one-step U-Net weights, we compute the mean and standard deviation (STD) of the changes in their norm and direction:

- Changes in the mean norm:

$$\Delta\eta_{\text{mean}} = \frac{1}{k}\sum_{i=1}^{k}\frac{\left|\eta^i_{\text{one-step}} - \eta^i_{\text{multi-step}}\right|}{\eta^i_{\text{multi-step}}} \tag{2}$$

- Changes in the mean direction:

$$\Delta\mathcal{V}_{\text{mean}} = \frac{1}{k}\sum_{i=1}^{k}\sqrt{\sum_{j=1}^{d}\left(\mathcal{V}^{i,j}_{\text{one-step}} - \mathcal{V}^{i,j}_{\text{multi-step}}\right)^2}. \tag{3}$$

- Changes in the STD norm:

$$\Delta\eta_{\text{std}} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}\left(\frac{\left|\eta^i_{\text{one-step}} - \eta^i_{\text{multi-step}}\right|}{\eta^i_{\text{multi-step}}} - \Delta\eta_{\text{mean}}\right)^2} \tag{4}$$

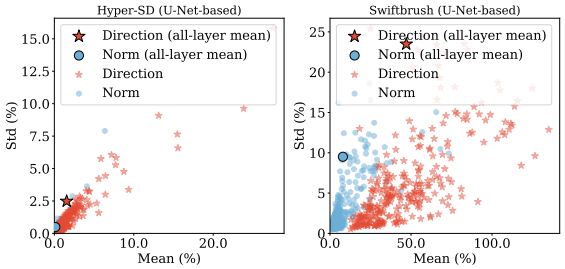



Figure A1. Visualization of changes in weight norm and direction (Hyper-SD [27] and Swiftbrush [25]).

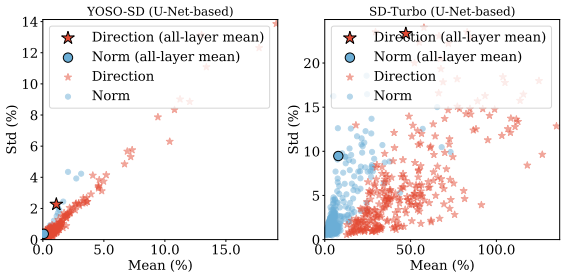



Figure A2. Visualization of changes in weight norm and direction (YOSO [22] and SD-Turbo [31]).

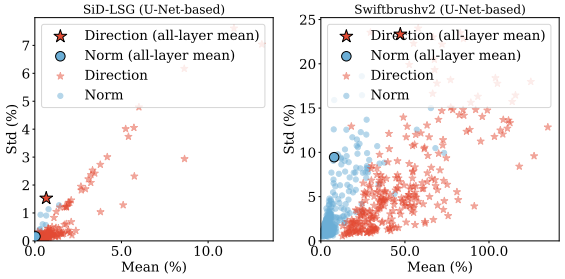



Figure A3. Visualization of changes in weight norm and direction (SiD-LSG [43] and Swiftbrushv2 [7]).

- Changes in the STD direction:

$$\Delta\mathcal{V}_{\text{std}} = \sqrt{\frac{1}{k}\left(\sum_{i=1}^{k}\sqrt{\sum_{j=1}^{d}\left(\mathcal{V}^{i,j}_{\text{one-step}} - \mathcal{V}^{i,j}_{\text{multi-step}}\right)^2} - \Delta\mathcal{V}_{\text{mean}}\right)^2} \tag{5}$$

As shown in Fig. A1, Fig. A2, Fig. A3, and Fig. A4, we also provide more visualization results including Swiftbrush [25], Hyper-SD [27], SD-Turbo [31], YOSO [22], SiD-LSG [43], Swiftbrushv2 [7], and PCM [35], etc. with SD 1.5, SD 2.1 or Pixart-$\alpha$ as the backbone. Similar conclusions can be drawn as in Sec. 1.

## F. Implementation Details

### F.1. Training and inference details

**Zero-shot generation.** 1) We implement WaDi using PyTorch and optimize it with the AdamW [20] optimizer ($\beta_1$ =

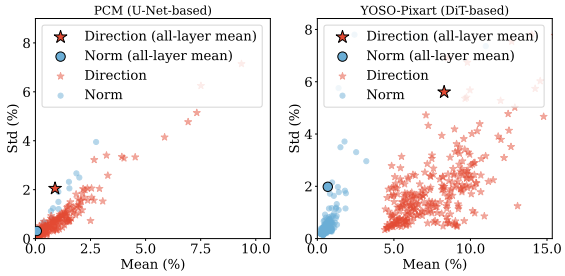



Figure A4. Visualization of changes in weight norm and direction (PCM [35] and YOSO-Pixart [22]).

0.9, $\beta_2 = 0.999$). LoRaD is applied to all linear layers in the U-Net/DiT architecture, including the feedforward layers, time_emb_proj layers, projection layers, and $Q, K, V$, out layers. Although we observe similar directional changes in convolutional layers, applying LoRaD to them introduces additional parameter overhead and lacks generality, as architectures like DiT [6] and autoregressive [34] models are primarily composed of linear layers. Notably, applying LoRaD solely to linear layers is sufficient to achieve SOTA performance. 2) We reproduce DMD2 [39], SiD-LSG [43], SwiftBrush [25], and PG-SB [26], following the default hyperparameter settings reported in their respective papers, except for using JourneyDB [33] as the training dataset. Specifically, our reproduction of DMD2 adopts the image-free variant without the second-stage GAN loss. For SiD-LSG, the guidance scales $k_1, k_2, k_3, k_4$ are set to 1.5.

**Baselines.** We conduct WaDi evaluations against a range of baselines on different model backbones. For the SD 1.5-based [28] backbone, we select LCM-LoRA [21], InstaFlow [19], UFOGen [38], DMD [40], DMD2 [39], SiD-LSG [43], PCM [35], Hyper-SD [27], and YOSO [22] as baselines. For the SD 2.1-based [28] backbone, we compare with SD-Turbo [31], SwiftBrush [25], SwiftBrushv2 [7], SiD-LSG [43], and TiUE [15]. For the DiT-based [6] backbone, SwiftBrush [25] and PG-SB [26] are chosen as baselines. In addition, we apply WaDi to four downstream tasks, including controllable generation, relation inversion, high-resolution synthesis, and image customization. For these tasks, we use ControlNet [41], Reversion [12], ScaleCrafter [10], DreamBooth [29], and LoRA [11] as baselines.

**Downstream tasks.** For all tasks except image personalization, we replace the original multi-step U-Net with our distilled one-step WaDi model to accelerate inference. For image customization, we apply LoRaD during fine-tuning. 1) During inference with ControlNet [41], Reversion [12], ScaleCrafter [10], and DreamBooth [29], we use the DDIM [32] scheduler with 50 inference steps. 2) In implementing DreamBooth, LoRA, and LoRaD, we adopt the prior-preservation loss with a weight factor of 1.0, and use the Adam [14] optimizer. The number of class-conditioned images is set to 200. For DreamBooth, the learning rate (LR) is set to 5e-6 with 800 training steps. For both LoRA and LoRaD, the LR is set to 1e-4, the number of training steps is 1500, and the rank is set to 64.

## F.2. More descriptions of downstream tasks.

Controllable generation introduces structured control signals (e.g., edge maps, human pose, or depth maps) into text-to-image models to guide the generation process toward producing images that not only match the textual description but also conform to specific spatial or semantic constraints. By incorporating external structural priors, controllable generation enables fine-grained manipulation of object layout, shape, and orientation—capabilities that are difficult to achieve with pure text prompts alone. Controllable generation plays an important role in tasks such as conditional image synthesis and creative visual design, where strict adherence to user-provided structure is crucial for usability and reliability.

Relation inversion aims to learn specific relational patterns (e.g., handshake, back-to-back, hugging) from a few input images and encode them into relation prompts, which can then be generalized to novel subjects, poses, and styles. Instead of focusing on individual object identity or appearance, relation inversion captures the spatial and semantic relationships between multiple entities, allowing the model to reproduce similar interactions under new contexts. Relation inversion enables controllable multi-subject generation and relational transfer.

High-resolution synthesis aims to generate high-resolution images (e.g., 512×512 or higher) with both strong global coherence and rich local detail. Compared to low-resolution generation, this task presents greater challenges in terms of spatial consistency, object fidelity, and fine-grained texture reconstruction. High-resolution synthesis requires the model to capture long-range dependencies across the image while preserving subtle local variations such as edges, patterns, and shading. Despite these challenges, it plays a crucial role in applications such as photo-realistic content creation and artistic image generation.

Image customization aims to learn the visual concept of a specific subject from a few example images and generalize it to new images. Specifically, it involves fine-tuning a pre-trained model to capture and retain the subject's distinctive visual features, enabling the model to generate personalized images that preserve the subject identity when given new text prompts. This capability is particularly valuable in applications such as creative design, personalized content generation, and artistic image synthesis.

**High-resolution synthesis.** ScaleCrafter [10] enables variable-resolution image generation in SD without retraining, which is accomplished by adjusting receptive fields within the pre-trained U-Net. To address the fixed-resolution

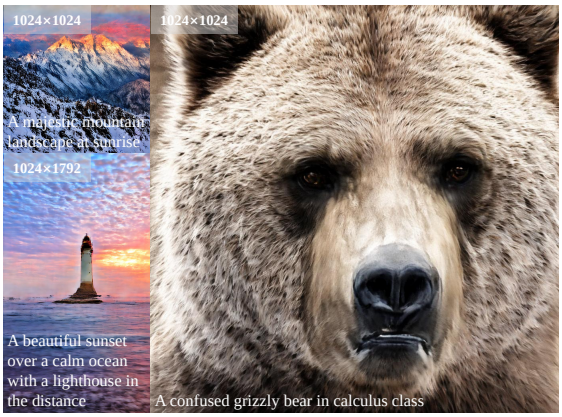



Figure A5. Quality results by ScaleCrafter [10] with WaDi.

**Algorithm 1** WaDi: Weight Direction-aware Distillation

**Require:** Pretrained real model $\epsilon_\psi$, fake model $\epsilon_{\phi_{\Theta^s}}$, one-step generator $G_{\lambda_{\Theta^l}}$, learning rates $\gamma_1$ and $\gamma_2$, initial timestep $t_{\text{init}}$, time-dependent weight function $\omega(t)$, prompts dataset $\mathcal{D}$, maximum number of timesteps $T_{\max}$ and scheduler schedule $\{(\alpha_t, \sigma_t)\}_{t=1}^{t=T}$

**Initialize:** $\phi_{\Theta^s} \leftarrow \psi$, $\lambda_{\Theta^l} \leftarrow \psi$, $\gamma_1 = 1e-4$, $\gamma_2 = 1e-2$, $t_{\text{init}} = 999$

**repeat**

  Sample input noise $\boldsymbol{z} \sim \mathcal{N}(0, I)$ and prompt $\boldsymbol{c} \sim \mathcal{D}$

  Generate one-step output $\boldsymbol{x}_0 = G_{\lambda_{\Theta^l}}(\boldsymbol{z}, \boldsymbol{c}, t_{\text{init}})$

  Sample timestep $t \sim \mathcal{U}(0.02T_{\max}, 0.98T_{\max})$ and noise $\epsilon \sim \mathcal{N}(0, I)$

  Compute noisy latent code $\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_0 + \sigma_t \epsilon$

  Update $G_{\lambda_{\Theta^l}}$ with $\lambda_{\Theta^l} \leftarrow \lambda_{\Theta^l} - \gamma_1 \left[\omega(t)\left(\epsilon_\psi(\boldsymbol{x}_t, \boldsymbol{c}, t) - \epsilon_{\phi_{\Theta^s}}(\boldsymbol{x}_t, \boldsymbol{c}, t)\right) \frac{\partial \boldsymbol{x}_0}{\partial \lambda_{\Theta^l}}\right]$

  Sample another timestep $\tilde{t} \sim \mathcal{U}(0.02T_{\max}, 0.98T_{\max})$ and noise $\tilde{\epsilon} \sim \mathcal{N}(0, I)$

  Compute noisy latent code $\boldsymbol{x}_{\tilde{t}} = \alpha_{\tilde{t}} \boldsymbol{x}_0 + \sigma_{\tilde{t}} \tilde{\epsilon}$

  Update $\epsilon_{\phi_{\Theta^s}}$ with $\Theta^s \leftarrow \Theta^s - \gamma_2 \nabla_{\phi_{\Theta^s}} \left\| \epsilon_{\phi_{\Theta^s}}(\boldsymbol{x}_{\tilde{t}}, \tilde{t}, \boldsymbol{c}) - \tilde{\epsilon} \right\|^2$

**until** processing 1.4M prompts or training budget is exhausted

**return** Trained one-step generator $G_{\lambda_{\Theta^l}}$

limitation inherited from SD 1.5/2.1, we integrate WaDi with ScaleCrafter. As illustrated in Fig. A5, this combination allows WaDi to generate high-fidelity images at varying resolutions, demonstrating strong scalability.

### F.3. Pseudo training code

As shown in Algorithm 1, we provide the pseudocode for WaDi to clearly outline the key steps of the algorithm.


Figure A6. User study examples.

### F.4. Comparison of training and inference time

We compared the performance of WaDi with other state-of-the-art (SOTA) methods on the zero-shot benchmark of COCO 2014. As shown in Tab. A2, WaDi demonstrates excellent inference efficiency, with an inference time of 0.11 seconds on both the SD 1.5 and SD 2.1 backbones, making it one of the fastest methods. In terms of training, WaDi completes distillation in only 2.1 A100 GPU days, significantly outperforming methods like InstaFlow [19] and DMD [40], which require much longer training times. WaDi achieves SOTA FID (10.79) and competitive CLIP score (0.31), striking a strong balance between speed and performance, particularly in image-free settings. This makes WaDi an efficient distillation solution, especially in environments with limited computational resources.

We measured the inference time of the models on a server equipped with an NVIDIA A40 GPU, using a batch size of 1. The experiments were conducted with PyTorch 2.4.0 and Hugging Face Diffusers 0.25.0, with the inference time including the computation of the text encoder and latent decoder.

### F.5. User study details

This study recruited 57 volunteers from our university to participate in a questionnaire-based evaluation. The questionnaire consisted of 44 questions, each presenting several images—one generated by our WaDi method and the others by alternative approaches, including SiD-LSG [43], DMD2 [39], Hyper-SD [27], YOSO [22], SwiftBrush [25], SwiftBrushv2 [7], ControlNet [41], Reversion [12], DreamBooth [29], LoRA [11], and ScaleCrafter [10]. An example of the questionnaire is shown in Fig. A6.

## G. Additional Results

### G.1. Results for HPSv2

Table A3 presents a quantitative comparison of WaDi with other SOTA methods on the HPSv2 benchmark. With an average score of 26.20, WaDi outperforms several competitive methods, including SiD-LSG [43] (26.11) and YOSO [22] (26.05). Notably, WaDi excels in the photo category, achieving a score of 26.80, demonstrating its strong text-to-image alignment capability. It is worth noting that Hyper-SD [27]

Table A2. Comparison of inference and training times of our method vs. other methods on the zero-shot benchmark of COCO 2014. * indicates our reproduced results, and † indicates results using the official pre-trained models. '-' denotes unknown. Best and second-best scores are in **bold** and underline, respectively.

| Method | NFEs | Type | Trainable params | FID ↓ | CLIP ↑ | Image-free? | Inference | A100 Days |
|---|---|---|---|---|---|---|---|---|
| Stable Diffusion 1.5-based backbone | | | | | | | | |
| SD 1.5 (*cfg* = 3.0) [28] | 25 | U-Net | 860M | 8.78 | 0.30 | ✗ | 1.11s | 4783 |
| LCM-LoRA [21]† | 1 | LoRA | 67.50 | 77.73 | 0.24 | ✗ | 0.11s | 1.3 |
| InstaFlow [19] | 1 | U-Net | 860M | 13.10 | 0.28 | ✗ | 0.11s | 183.2 |
| UFOGen [38] | 1 | U-Net | 860M | 12.78 | - | ✗ | - | - |
| DMD [40] | 1 | U-Net | 860M | 11.49 | **0.32** | ✗ | 0.11s | 108 |
| DMD2 [39]* | 1 | U-Net | 860M | 12.96 | 0.30 | ✓ | 0.11s | 5.1 |
| SiD-LSG [43]* | 1 | U-Net | 860M | 14.27 | 0.30 | ✓ | 0.11s | 6.4 |
| PCM [35] | 1 | U-Net | 860M | 17.91 | 0.29 | ✗ | - | unk |
| Hyper-SD [27]† | 1 | LoRA | 67.25M | 22.90 | 0.31 | ✗ | 0.11s | 33.3 |
| YOSO [22]† | 1 | LoRA | 67.25M | 23.68 | 0.29 | ✗ | 0.11s | 20 |
| WaDi | 1 | LoRaD | 83.80M | **10.79** | 0.31 | ✓ | 0.11s | 2.1 |
| Stable Diffusion 2.1-based backbone | | | | | | | | |
| SD 2.1 (*cfg* = 3.0) [28] | 25 | U-Net | 865M | 9.60 | 0.32 | ✗ | 1.04s | 8332 |
| SD-Turbo [31]† | 1 | U-Net | 865M | 16.14 | **0.33** | ✗ | 0.11s | - |
| Swiftbrush [25] | 1 | U-Net | 865M | 16.67 | 0.29 | ✓ | 0.11s | 4.1 |
| Swiftbrushv2 [7]* | 1 | U-Net+LoRA | 884.14M | 15.98 | **0.33** | ✓ | 0.11s | 24.1 |
| SiD-LSG [43]* | 1 | U-Net | 865M | 15.17 | 0.30 | ✓ | 0.11s | 6.4 |
| TiUE [15]† | 1 | U-Net | 865M | 13.49 | 0.31 | ✓ | 0.16s | 3.9 |
| WaDi | 1 | LoRaD | 94.43M | **12.34** | 0.31 | ✓ | 0.11s | 2.1 |
| PixArt-$\alpha$-based backbone $256 \times 256$ | | | | | | | | |
| PixArt-$\alpha$ (*cfg* = 4.5) [6]† | 20 | DiT | 0.6B | 8.75 | 0.32 | ✗ | 0.59s | 753 |
| Swiftbrush [25]* | 1 | DiT | 0.6B | 29.89 | 0.28 | ✓ | 0.05s | 2.6 |
| PG-SB [26]* | 1 | DiT | 0.6B | 25.58 | 0.28 | ✓ | 0.05s | 2.6 |
| WaDi | 1 | LoRaD | 81.22M | **18.99** | **0.30** | ✓ | 0.05s | 1.6 |

Table A3. Quantitative comparison of WaDi and other methods on HPSv2 results. * indicates our reproduced results, and † indicates results using the official pre-trained models. '-' denotes unknown. Best and second-best scores are in **bold** and underline, respectively.

| Method | Anime | Photo | Concept Art | Paintings | Average |
|---|---|---|---|---|---|
| Stable Diffusion 1.5-based backbone | | | | | |
| SD 1.5 [28] | 26.51 | 27.19 | 26.06 | 26.12 | 26.47 |
| InstaFlow [19] | 26.10 | 26.62 | 25.92 | 25.95 | 26.15 |
| DMD2 [39]* | 25.65 | 26.13 | 24.98 | 25.22 | 25.49 |
| SiD-LSG [43]* | 26.24 | 26.46 | 25.88 | 25.86 | 26.11 |
| Hyper-SD [27]† | **27.37** | **27.59** | **27.13** | **27.15** | **27.31** |
| YOSO [22]† | 26.24 | 26.26 | 25.92 | 25.79 | 26.05 |
| WaDi | 26.39 | 26.80 | 25.79 | 25.81 | 26.20 |

achieves SOTA performance across all metrics, thanks to its use of the aesthetic predictor of LAION dataset, the ImageReward [37] aesthetic preference reward model, and the SOLO [36] visual perception model, which guide the optimization process through multiple supervision signals.

## G.2. Additional ablation studies

In our main experiments, we apply LoRaD only to linear layers to achieve a better trade-off between performance and parameter efficiency. As shown in Tab. A4, extending LoRaD to convolutional layers leads to a performance drop, suggesting that LoRaD already possesses sufficient representational capacity. Extending to more layers increases parameter count and may introduce overfitting.

Table A4. Ablation experiments on the impact of LoRaD application layer types.

| Type | #Trainable Params | FID | CLIP |
|---|---|---|---|
| Linear | 83.8M | **10.79** | **0.31** |
| Linear + Conv | 174.89M | 16.42 | 0.30 |

To ensure the student initially matches the teacher, we initialize the student network with teacher weights. Specifically, in LoRaD, we set the low-rank matrix $A = 0$ and initialize $B$ with Xavier, resulting in $AB = 0$ at the start of training—thus applying no rotation and preserving the teacher's parameter directions. As shwon in Tab. A4, we also experimented with Xavier initialization for both $A$ and $B$, which led to a significantly worse FID (10.79 → 18.41), indicating degraded convergence. This may be because non-zero initialization perturbs the pretrained model and causes

Table A5. Ablation experiments on LoraD initialization strategy.

| Initialization | FID | CLIP |
|---|---|---|
| $A = 0,\ B =$ Xavier | **10.79** | **0.31** |
| $A =$ Xavier, $B =$ Xavier | 18.41 | 0.31 |

optimization to converge to a suboptimal region, thereby affecting final performance.

Our zero initialization follows recent work. For example, both LoRA [11] and DoRA [18] set at the beginning of training, making the model initially equivalent to the pretrained weights , thus avoiding disruption of the original model behavior. Similarly, ControlNet 41 initializes the image-conditional branch to output zero, ensuring that the initial behavior remains consistent with the base model and allowing conditional control signals to be gradually introduced through training. This prevents training instability or performance degradation caused by the premature influence of untrained control branches. We believe that pretrained weights provide a strong anchor for distillation models, making optimization more stable and convergence easier. In addition, recent work PiSSA [24] is the first to apply SVD to the original model, leveraging principal singular values and vectors to initialize the adapter for fine-tuning. This approach further accelerates LoRA's convergence and improves its performance. These observations collectively highlight the crucial role of good initialization in achieving both fast convergence and strong final performance. Motivated by this, we plan to further investigate initialization strategies for LoRaD in future work.

### G.3. More quantitative results

We compare WaDi with representative distillation methods on COCO2014 [16], and further evaluate its generalization on COCO2017 [16]. As shown in Tab. A6, WaDi consistently achieves strong performance across three backbone models: SD 1.5, SD 2.1, and PixArt-$\alpha$. Specifically, WaDi achieves the best or second-best results in both FID and CLIP scores across all settings, and strikes a favorable balance between precision and recall, demonstrating strong capability in image quality and semantic alignment. Notably, WaDi is distilled using only 1.4M text prompts, yet outperforms or matches methods that rely on over 3M prompts, such as LCM [21] (12M) and YOSO [22] (4M). This highlights both the efficiency of WaDi under low-resource settings and the effectiveness of its distillation mechanism. In contrast to methods like Hyper-SD [27], YOSO [22], and SD-Turbo [31], WaDi requires no real images and is trained via prompt-only distillation, enhancing its practicality and scalability.

### G.4. Convergence analysis

Fig A7 presents the convergence analysis of WaDi compared to other SOTA one-step distillation models, including SwiftBrush [25], SiD-LSG [43], and DMD2 [39]. The plot shows that WaDi achieves faster convergence, with both FID and CLIP scores improving consistently across iterations. WaDi demonstrates superior performance in terms of FID reduction, reaching a lower value than the other models by the end of the training. In terms of CLIP, WaDi maintains a steady and significant increase, outperforming SwiftBrush [25] and SiD-LSG [43] in the later stages. This highlights WaDi's efficiency in both training stability and perceptual alignment, which aligns with the qualitative results in Fig. A8.

### G.5. More visualization results

Fig. A9 to Fig. A16 present the sampling results of WaDi (based on SD 1.5), the sampling results of WaDi (based on PixArt-$\alpha$), additional qualitative comparisons, visualizations of ControlNet-WaDi and Reversion-WaDi, qualitative results of DreamBooth-LoRaD, and extended visualizations of ScaleCrafter-WaDi, further demonstrating the generality and adaptability of our approach across diverse tasks.

## References


[1] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008. 1

[2] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *ICML*, pages 254–263. PMLR, 2018. 2

[3] Alessandro Barp, Chris J Oates, Emilio Porcu, and Mark Girolami. A riemann–stein kernel method. *Bernoulli*, 28(4): 2181–2208, 2022. 2

[4] Hadi Beik-Mohammadi, Søren Hauberg, Georgios Arvanitidis, Gerhard Neumann, and Leonel Rozo. Reactive motion generation on learned riemannian manifolds. *The International Journal of Robotics Research*, 42(10):729–754, 2023. 2

[5] Nicolas Boumal. *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023. 1

[6] Junsong Chen, YU Jincheng, GE Chongjian, Lewei Yao, Enze Xie, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-$\alpha$: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *ICLR*, 2023. 3, 5, 7

[7] Trung Dao, Thuan Hoang Nguyen, Thanh Le, Duc Vu, Khoi Nguyen, Cuong Pham, and Anh Tran. Swiftbrush v2: Make your one-step diffusion model better than its teacher. In *ECCV*, pages 176–192. Springer, 2024. 2, 3, 4, 5, 7

[8] Yanhong Fei, Yingjie Liu, Chentao Jia, Zhengyu Li, Xian Wei, and Mingsong Chen. A survey of geometric optimization for deep learning: from euclidean space to riemannian manifold. *CSUR*, 57(5):1–37, 2025. 2

[9] Andi Han, Bamdev Mishra, Pratik Kumar Jawanpuria, and Junbin Gao. On riemannian optimization over positive definite

Table A6. Quantitative comparison of WaDi and other methods on zero-shot COCO 2017 results. * indicates our reproduced results, and † indicates results using the official pre-trained models. '-' denotes unknown. Best and second-best scores are in **bold** and underline, respectively.

| Method | #Params | NFEs | Type | Trainable params | FID ↓ | CLIP ↑ | Precision ↑ | Recall ↑ | Image-free? | Training Data |
|---|---|---|---|---|---|---|---|---|---|---|
| Stable Diffusion 1.5-based backbone | | | | | | | | | | |
| SD 1.5 ($cfg = 3.0$) [28] | 860M | 25 | U-Net | 860M | 19.80 | 0.31 | 0.64 | 0.60 | ✗ | 5B |
| LCM-LoRA [21]† | 860M | 1 | LoRA | 67.50M | 89.65 | 0.24 | 0.22 | 0.24 | ✗ | 12M |
| InstaFlow [19] | 860M | 1 | U-Net | 860M | 23.49 | **0.31** | 0.53 | 0.46 | ✗ | 3.2M |
| UFOGen [38] | 860M | 1 | U-Net | 860M | 22.50 | **0.31** | - | - | ✗ | 12M |
| DMD2 [39]* | 860M | 1 | U-Net | 860M | 23.30 | 0.30 | 0.60 | 0.49 | ✓ | 1.4M |
| SiD-LSG [43]* | 860M | 1 | U-Net | 860M | 24.22 | 0.30 | 0.60 | 0.52 | ✓ | 1.4M |
| Hyper-SD [27]† | 860M | 1 | LoRA | 67.25M | 32.49 | **0.31** | 0.52 | 0.33 | ✗ | - |
| YOSO [22]† | 860M | 1 | LoRA | 67.25M | 33.54 | 0.29 | 0.50 | 0.44 | ✗ | 4M |
| WaDi | 860M | 1 | LoRaD | 83.80M | **20.86** | **0.31** | **0.63** | **0.54** | ✓ | 1.4M |
| Stable Diffusion 2.1-based backbone | | | | | | | | | | |
| SD 2.1 ($cfg = 3.0$) [28] | 865M | 25 | U-Net | 865M | 19.66 | 0.32 | 0.66 | 0.57 | ✗ | 5B |
| SD-Turbo [31]† | 865M | 1 | U-Net | 865M | 26.36 | **0.34** | **0.69** | 0.47 | ✗ | - |
| Swiftbrush [25] | 865M | 1 | U-Net | 865M | 26.87 | 0.32 | 0.61 | 0.44 | ✓ | 1.4M |
| Swiftbrushv2 [7]* | 865M | 1 | U-Net+LoRA | 884.14M | 25.96 | 0.33 | 0.65 | 0.45 | ✓ | 3.3M |
| SiD-LSG [43]* | 865M | 1 | U-Net | 865M | 25.02 | 0.30 | 0.62 | 0.51 | ✓ | 1.4M |
| WaDi | 865M | 1 | LoRaD | 94.43M | **22.62** | 0.31 | 0.65 | **0.53** | ✓ | 1.4M |
| PixArt-$\alpha$-based backbone | | | | | | | | | | |
| PixArt-$\alpha$ ($cfg = 4.5$) [6]† | 0.6B | 20 | DiT | 0.6B | 20.85 | 0.27 | 0.65 | 0.59 | ✗ | 25M |
| Swiftbrush [25]* | 0.6B | 1 | DiT | 0.6B | 41.07 | 0.28 | 0.53 | 0.35 | ✓ | 1.4M |
| PG-SB [26]* | 0.6B | 1 | DiT | 0.6B | 35.84 | 0.28 | 0.57 | 0.36 | ✓ | 1.4M |
| WaDi | 0.6B | 1 | LoRaD | 81.22M | **28.91** | **0.30** | **0.62** | **0.37** | ✓ | 1.4M |

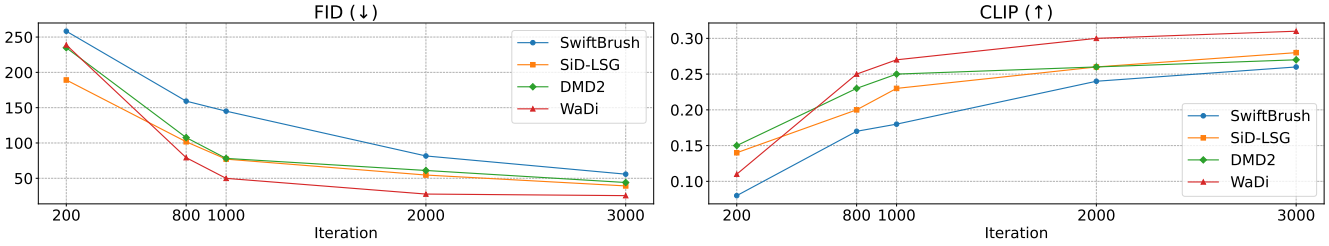
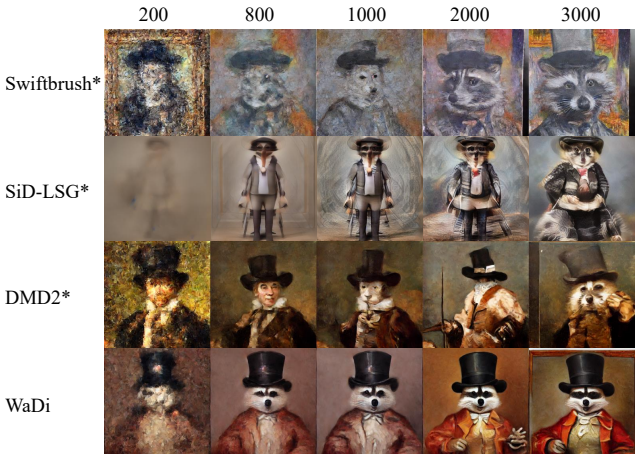



Figure A7. Convergence analysis of WaDi and other methods. * indicates our reproduced results.




"A racoon wearing formal clothes, wearing a tophat. Oil painting in the style of Rembrandt"

Figure A8. Iteration qualitative results. "* indicates our reproduced results"

Figure A9. One-step generated images using our proposed method WaDi (SD 1.5).

matrices with the bures-wasserstein geometry. *NeurIPS*, 34: 8940–8953, 2021. 2

[10] Yingqing He, Shaoshu Yang, Haoxin Chen, Xiaodong Cun,

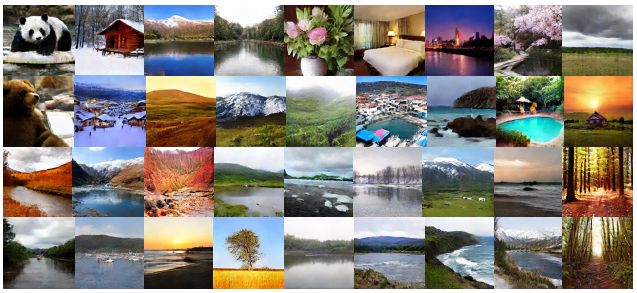


Figure A10. One-step generated images using our proposed method WaDi (PixArt-$\alpha$ $256 \times 256$).

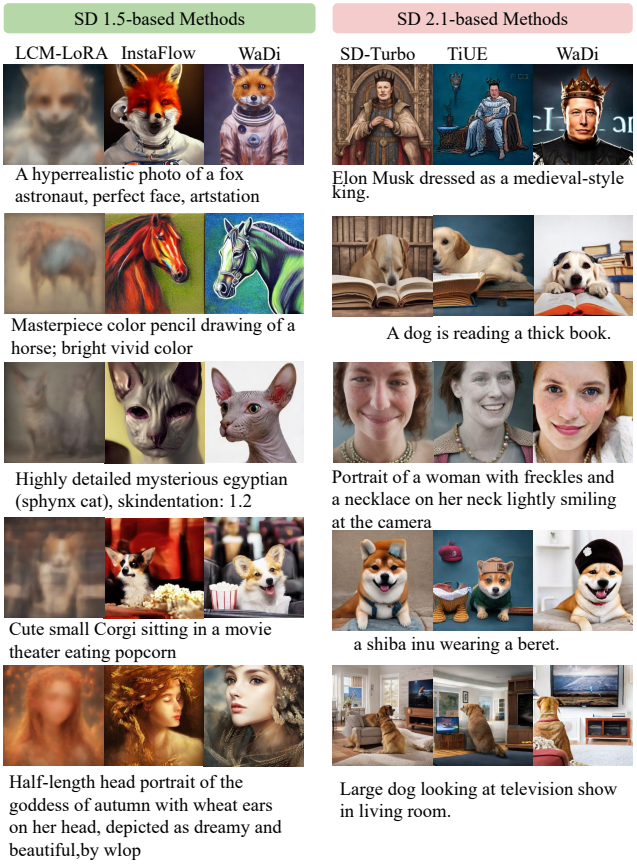



Figure A11. Qualitative comparison with other methods.


Menghan Xia, Yong Zhang, Xintao Wang, Ran He, Qifeng Chen, and Ying Shan. Scalecrafter: Tuning-free higher-resolution visual generation with diffusion models. In *ICLR*, 2023. 3, 4, 12

[11] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 3, 4, 6

[12] Ziqi Huang, Tianxing Wu, Yuming Jiang, Kelvin CK Chan, and Ziwei Liu. Reversion: Diffusion-based relation inversion from images. In *SIGGRAPH Asia*, pages 1–11, 2024. 3, 4, 10

[13] Jean Kaddour, Linqing Liu, Ricardo Silva, and Matt J Kusner. When do flat minima optimizers work? *NeurIPS*, 35:16577–16595, 2022. 2

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[15] Senmao Li, Lei Wang, Kai Wang, Tao Liu, Jiehang Xie, Joost van de Weijer, Fahad Shahbaz Khan, Shiqi Yang, Yaxing Wang, and Jian Yang. One-way ticket: Time-independent unified encoder for distilling text-to-image diffusion models. In *CVPR*, pages 23563–23574, 2025. 3, 5

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014. 6

[17] Chang Liu and Jun Zhu. Riemannian stein variational gradient descent for bayesian inference. In *AAAI*, 2018. 2

[18] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *ICML*, 2024. 2, 6

[19] Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, et al. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *ICLR*, 2023. 3, 4, 5, 7

[20] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 2

[21] Simian Luo, Yiqin Tan, Suraj Patil, Daniel Gu, Patrick von Platen, Apolinário Passos, Longbo Huang, Jian Li, and Hang Zhao. Lcm-lora: A universal stable-diffusion acceleration module. *arXiv preprint arXiv:2311.05556*, 2023. 3, 5, 6, 7

[22] Yihong Luo, Xiaolong Chen, Xinghua Qu, Tianyang Hu, and Jing Tang. You only sample once: Taming one-step text-to-image synthesis by self-cooperative diffusion gans. *arXiv preprint arXiv:2403.12931*, 2024. 2, 3, 4, 5, 6, 7

[23] Kaifeng Lyu, Zhiyuan Li, and Sanjeev Arora. Understanding the generalization benefit of normalization layers: Sharpness reduction. *NeurIPS*, 35:34689–34708, 2022. 2

[24] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *NeurIPS*, 37:121038–121072, 2024. 6

[25] Thuan Hoang Nguyen and Anh Tran. Swiftbrush: One-step text-to-image diffusion model with variational score distillation. In *CVPR*, pages 7807–7816, 2024. 2, 3, 4, 5, 6, 7

[26] Viet Nguyen, Anh Nguyen, Trung Dao, Khoi Nguyen, Cuong Pham, Toan Tran, and Anh Tran. Snoopi: Supercharged one-step diffusion distillation with proper guidance. *arXiv preprint arXiv:2412.02687*, 2024. 3, 5, 7

[27] 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 *NeurIPS*, 2024. 2, 3, 4, 5, 6, 7

[28] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022. 3, 5, 7

[29] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*, pages 22500–22510, 2023. 3, 4, 11

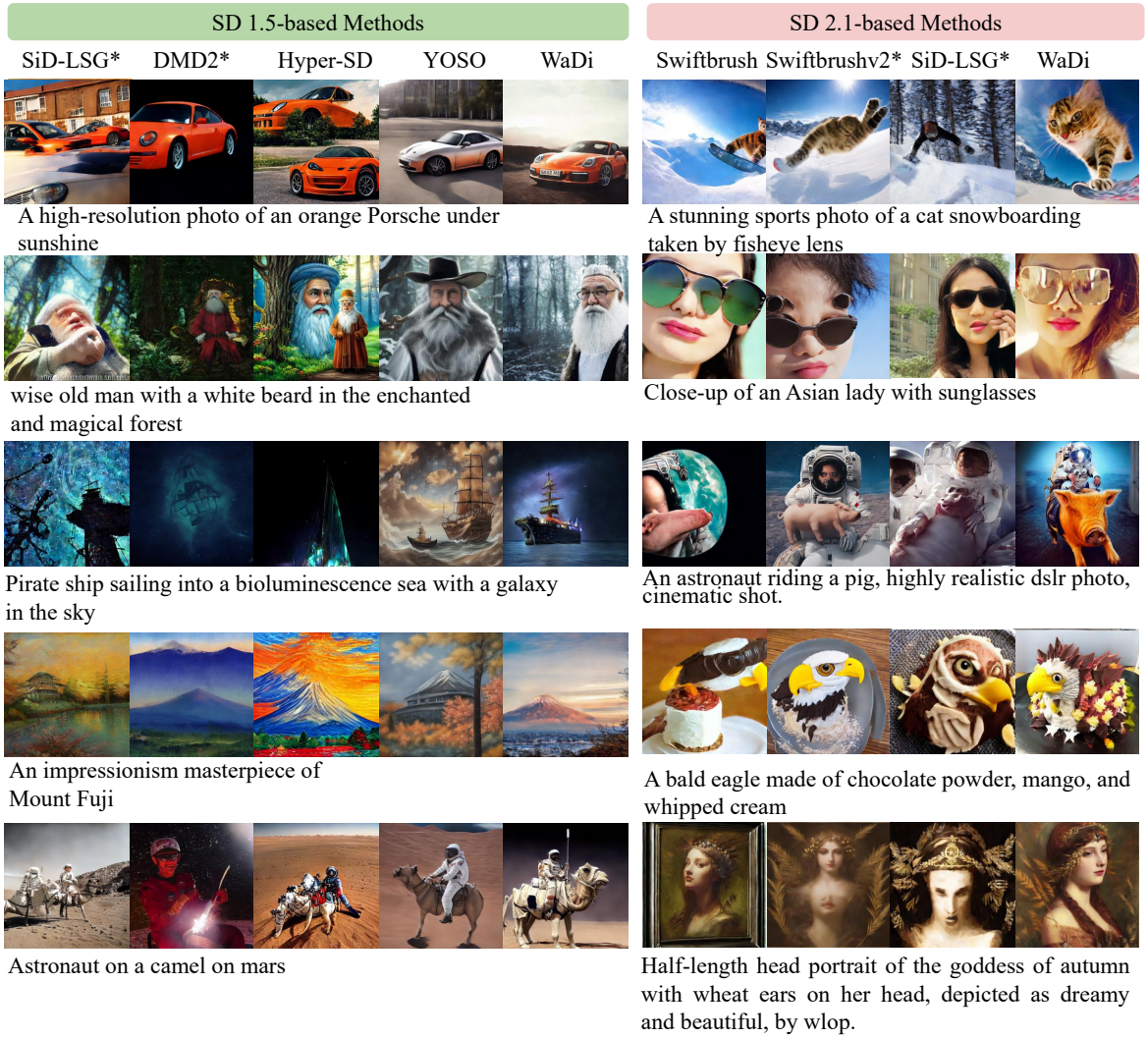



Figure A12. Qualitative comparison to state-of-the-art one-step distillation models. * indicates our reproduced results.


[30] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *NeurIPS*, 29, 2016. 2

[31] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *ECCV*, pages 87–103. Springer, 2024. 2, 3, 5, 6, 7

[32] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 3

[33] Keqiang Sun, Junting Pan, Yuying Ge, Hao Li, Haodong Duan, Xiaoshi Wu, Renrui Zhang, Aojun Zhou, Zipeng Qin, Yi Wang, et al. Journeydb: A benchmark for generative image understanding. *NeurIPS*, 36:49659–49678, 2023. 3

[34] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024. 3

[35] Fu-Yun Wang, Zhaoyang Huang, Alexander Bergman, Dazhong Shen, Peng Gao, Michael Lingelbach, Keqiang Sun, Weikang Bian, Guanglu Song, Yu Liu, et al. Phased consistency models. *NeurIPS*, 37:83951–84009, 2024. 2, 3, 5

[36] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. In *ECCV*, pages 649–665. Springer, 2020. 5

[37] 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. *NeurIPS*, 36:15903–15935, 2023. 5

[38] Yanwu Xu, Yang Zhao, Zhisheng Xiao, and Tingbo Hou. Ufogen: You forward once large scale text-to-image generation via diffusion gans. In *CVPR*, pages 8196–8206, 2024. 3, 5, 7

[39] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and Bill Freeman. Improved distribution matching distillation for fast image synthesis. *NeurIPS*, 37:47455–47487, 2024. 3, 4, 5, 6, 7

[40] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shecht-

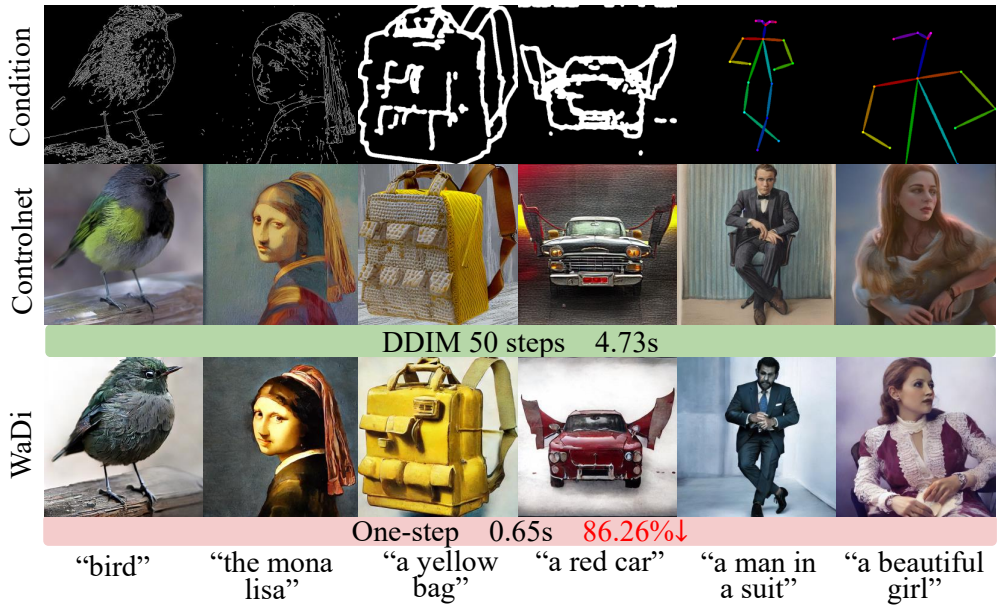



Figure A13. Quality results by Controlnet [42] with or without WaDi.

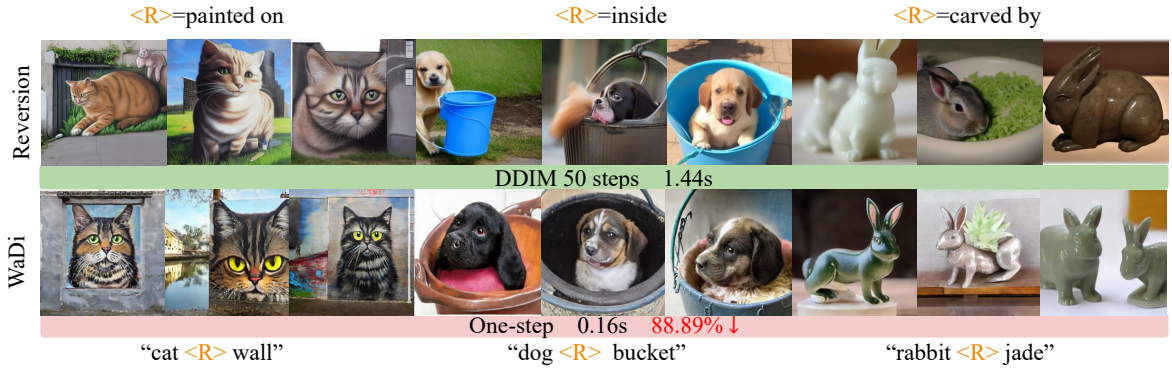



Figure A14. Quality results by Reversion [12] with or without WaDi.

man, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *CVPR*, pages 6613–6623, 2024. 3, 4, 5

[41] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, pages 3836–3847, 2023. 3, 4, 6

[42] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *NeurIPS 2022 Workshop on Score-Based Methods*, 2022. 10

[43] Mingyuan Zhou, Zhendong Wang, Huangjie Zheng, and Hai Huang. Long and short guidance in score identity distillation for one-step text-to-image generation. *arXiv preprint arXiv:2406.01561*, 2024. 2, 3, 4, 5, 6, 7

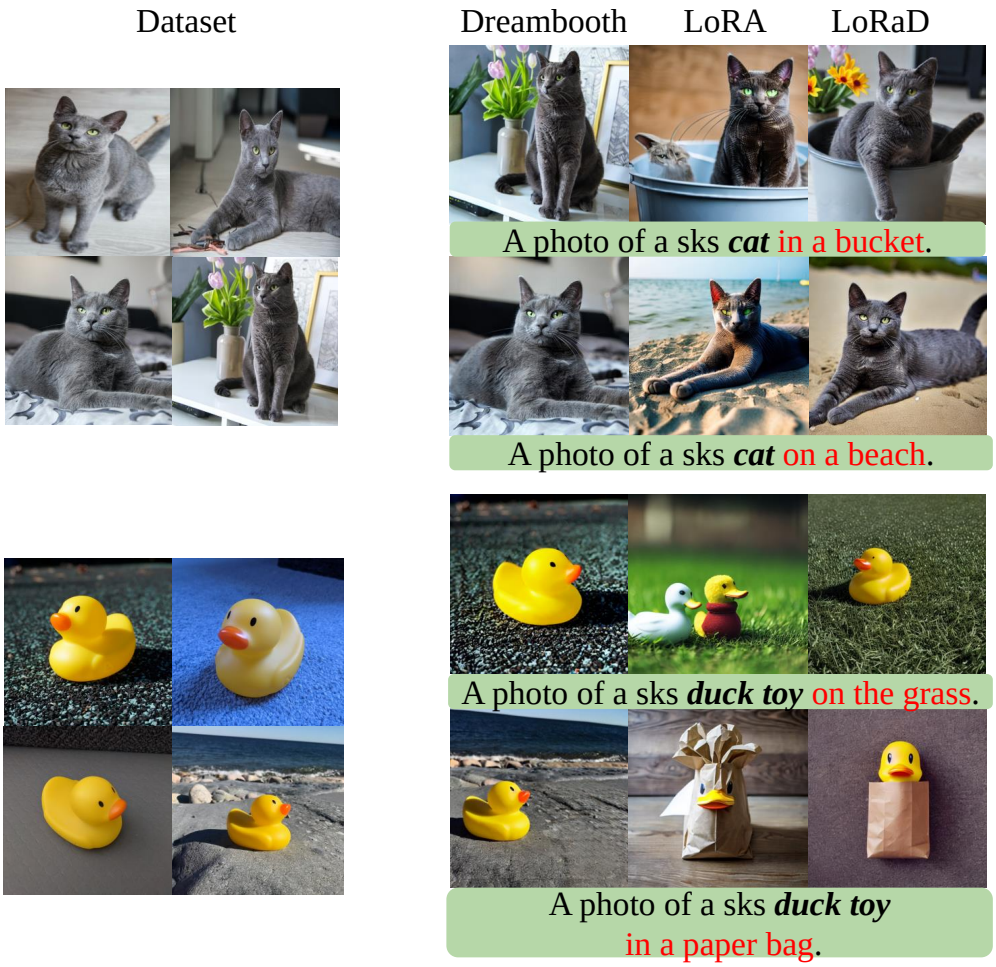



Figure A15. Quality results by Dreambooth [29] with or without LoRaD.

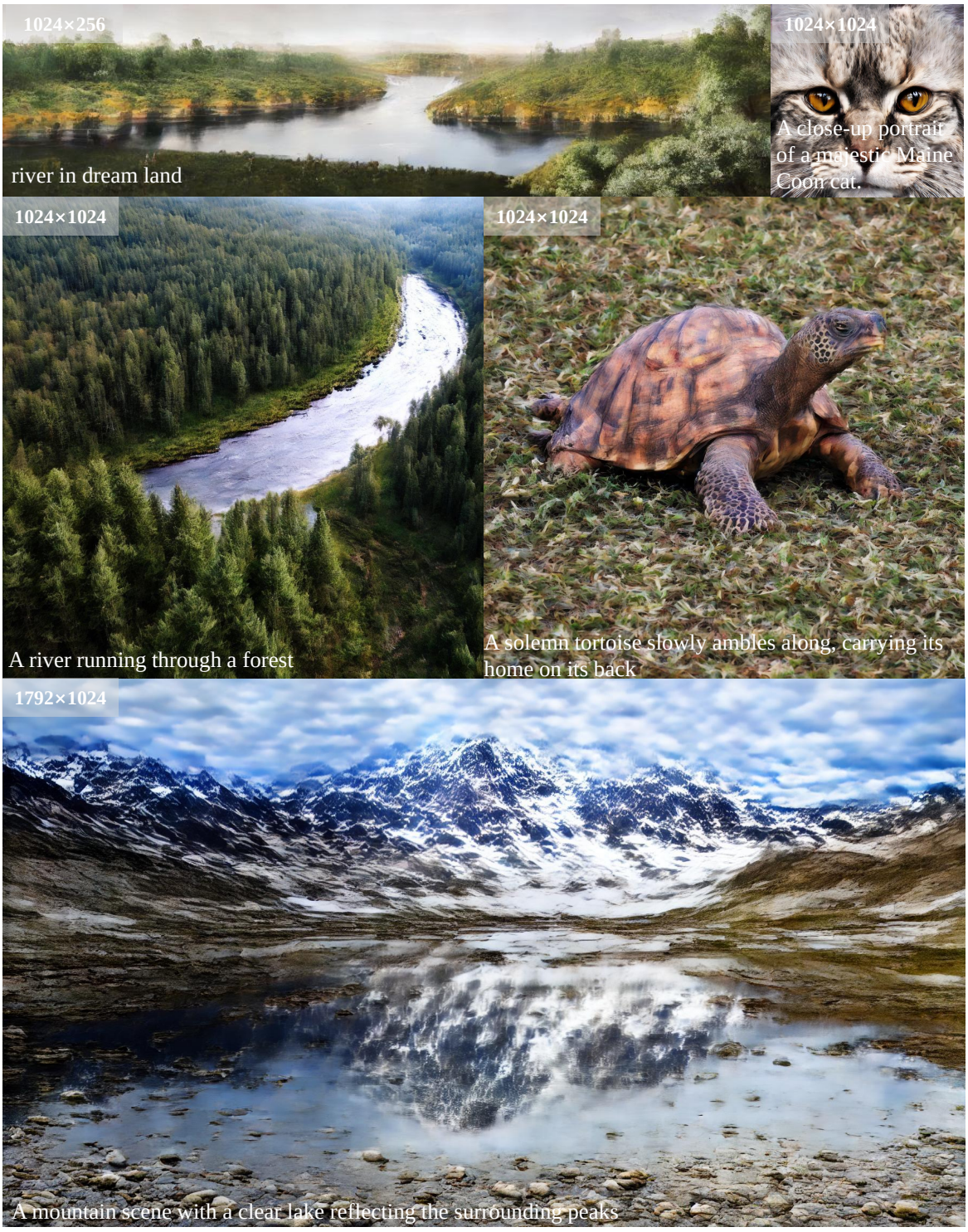



Figure A16. Quality results by ScaleCrafter [10] with WaDi.