One-Way Ticket : Time-Independent Unified Encoder for Distilling Text-to-Image Diffusion Models

— Appendix —

We provide implementation details (see Appendix A) and additional results (see Appendix B) for our *TiUE* image generation method with **loop-free** inference. Subsequently, we discuss the limitations and future work (see Appendix C), broader impacts (see Appendix D), ethical statement (see Appendix E), and reproducibility statement (see Appendix F).

A. Implementation Details

A.1. Evaluation Datasets and Metrics

Datasets. We conduct comparisons on four datasets to evaluate the density of image generation: AFHQ [6], CelebA-HQ [27], DrawBench [61], and PartiPrompts [81]. Since the AFHQ and CelebA-HQ datasets contain animals and human faces respectively, we utilize text prompts with the format: "a photo of <cat/dog/wild animal>" and "a photo of <man/woman>".

Metrics. We leverage code from the popular GitHub repository "StudioGAN" [23–25] [2] to calculate three metrics: Precision Recall [32], Density, and Coverage [52]. For FID [17] and Clipscore [16] metrics, we employ the official envaluation code from GigaGAN [26] [3]

A.2. Baseline Implementations

We use the official implementation of Instaflow [40] [4] , LCM [44] [5] , SD-Turbo [64] [6] , and SwiftBrush [53][7]. For SwiftBrushv2 [7], we re-implemented the work using the same amount of training data and computational resources as our method. All experiments are conducted at a standard resolution of 512×512 pixels on a single 3090 GPU device.

A.3. Training Details

We use Stable Diffusion 2.1 (SD 2.1)[8] to initialize the teacher SD generator and SD-LoRA generator, and the student generator. We implement our method with PyTorch, and use the Adam optimizer [30] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ to train both the student generator and SD-LoRA generator. When calculating the VSD loss $\mathcal{L}_{vsd}$, we use classifier-free guidance with a value of 4.5 like SwiftBrush [53] for both the teacher SD generator and the SD-LoRA generator.

JourneyDB Datasets. In JourneyDB datasets [54], there are 4M (4,189,737) captions in the training sets. We remove duplicate captions from the training set, leaving 1,418,723 unique captions. These captions are used as prompts to train the student generator. We train our model on NVIDIA 8×A40 48G GPU with batch size 64 and take 3 epochs.

A.4. The Green and Red Arrows in Fig. 4 and Fig. 5

The green arrows indicate the skip connections that transfer features from the middle layers of the encoder to the corresponding decoder layers, while the red arrows represent the path where features from the final encoder layer are inputs to the decoder.

At each time step, the decoder receives both the Mid-Block outputs and skip-connection features from the encoder. Since the Mid-Block does not receive skip-connection features, it is not shown in Fig. 4 and Fig. 5 and is considered part of the encoder.

A.5. Meaning of Loop-Free

We regard iterative denoising in the vanilla multi-step DMs as a "loop" process, while ours does not require any iterative process. Our method denoises in parallel with 4 decoder steps, achieving "loop-free" generation.

We further explain Eq. (5) in main paper and provide a mathematical interpretation of the 1-step inference for our 1-step encoder and 4-step decoder (i.e., K=4) design. The student generator ϵ_θ^{SG} takes as input a random noise ϵ, also referred to as z_K. As shown in Fig. 5, we only need to calculate skip connections and output features of the UNet-Encoder in the initiation step (t=K) as: $f = \epsilon_\theta^{SG\text{-}EN}(\epsilon, K, y)$. Then, the predicted noise of UNet-Decoder at step t (t=4, 3, 2, 1) can be calculated as $\epsilon^t = \epsilon_\theta^{SG\text{-}DE}(f, t, y)$ in parallel. Using the DDIM scheduler, the latent at step t can be written as:

$$z_3 = \sqrt{\frac{\alpha_3}{\alpha_4}}\epsilon + \sqrt{\alpha_3}\left(\sqrt{\frac{1}{\alpha_3}-1} - \sqrt{\frac{1}{\alpha_4}-1}\right)\cdot\epsilon^4, \quad (7)$$

$$z_2 = \sqrt{\frac{\alpha_2}{\alpha_3}}z_3 + \sqrt{\alpha_2}\left(\sqrt{\frac{1}{\alpha_2}-1} - \sqrt{\frac{1}{\alpha_3}-1}\right)\cdot\epsilon^3, \quad (8)$$

$$z_1 = \sqrt{\frac{\alpha_1}{\alpha_2}}z_2 + \sqrt{\alpha_1}\left(\sqrt{\frac{1}{\alpha_1}-1} - \sqrt{\frac{1}{\alpha_2}-1}\right)\cdot\epsilon^2, \quad (9)$$

[2] https://github.com/POSTECH-CVLab/PyTorch-StudioGAN
[3] https://github.com/lucidrains/gigagan-pytorch
[4] https://github.com/gnobitab/InstaFlow
[5] https://latent-consistency-models.github.io/
[6] https://github.com/Stability-AI/generative-models
[7] https://github.com/VinAIResearch/SwiftBrush
[8] https://huggingface.co/stabilityai/stable-diffusion-2-1-base

Dataset	Base Model	Step	Param	COCO2014-30K					COCO2017-5K					Inference↓		Training Data		A100
Method \ Metrics				FID↓	CLIP↑	Precision↑	Recall↑	F1↑	FID↓	CLIP↑	Precision↑	Recall↑	F1↑	Time (ms)	Memory (GB)	Size↓	Image Free	Days↓
SD1.5 [58] (*cfg*=7.5)†	–	50	860M	16.08	0.325	0.717	0.527	0.607	23.39	0.326	0.776	0.587	0.668	2503.0	4.04	5B	✗	4783
SD1.5 [58] (*cfg*=4.5)†	–	50	860M	9.90	0.322	0.727	0.585	0.648	19.87	0.323	0.764	0.649	0.702	2503.0	4.04	5B	✗	4783
SD2.1 [58] (*cfg*=7.5)†	–	50	865M	16.10	0.328	0.723	0.489	0.583	25.40	0.328	0.769	0.561	0.649	2244.2	3.89	5B	✗	8332
SD2.1 [58] (*cfg*=4.5)†	–	50	865M	12.22	0.325	0.734	0.526	0.614	22.24	0.298	0.788	0.606	0.685	2244.2	3.89	5B	✗	8332
FasterD [34] (*cfg*=7.5)†	SD1.5	50	860M	12.93	0.326	0.693	0.532	0.601	23.10	0.325	0.687	0.601	0.641	1476.0	21.83	–	–	–
FasterD [34] (*cfg*=4.5)†	SD1.5	50	860M	12.05	0.323	0.672	0.569	0.617	22.32	0.322	0.670	0.638	0.654	1476.0	21.83	–	–	–
FasterD [34] (*cfg*=7.5)†	SD2.1	50	865M	13.64	0.329	0.708	0.512	0.594	23.65	0.329	0.698	0.572	0.629	1356.0	21.26	–	–	–
FasterD [34] (*cfg*=4.5)†	SD2.1	50	865M	12.42	0.326	0.699	0.551	0.616	22.61	0.325	0.707	0.616	0.659	1356.0	21.26	–	–	–
GigaGAN [26]*	GAN	1	1.0B	9.24	0.325	0.724	0.547	0.623	–	–	–	–	–	–	–	2.7B	✗	6250
InstaFlow [40]†	SD1.5	1	0.9B	13.78	0.288	0.654	0.521	0.580	**19.00**	0.293	0.729	0.613	0.666	111.3	3.99	3.2M	✗	183.2
LCM [44]†	SD1.5	1	860M	132.09	0.230	0.109	0.194	0.140	143.73	0.229	0.118	0.291	0.168	236.2	5.88	12M	✗	1.3
LCM-LoRA [45] †	SD1.5	1	860M	115.21	0.280	0.069	0.221	0.105	126.82	0.280	0.070	0.265	0.111	101.4	4.66	12M	✗	1.3
Hyper-SD [57] †	SD1.5	1	860M	20.90	0.325	0.743	0.324	0.451	30.45	0.325	0.799	0.424	0.554	117.5	4.54	unk.	✗	33.3
SD-Turbo [64]†	SD2.1	1	865M	19.51	0.331	0.758	0.458	0.571	29.35	0.331	0.786	0.445	0.568	140.0	3.86	unk.	✗	unk.
TCD [86]†	SD2.1	1	865M	68.01	0.301	0.234	0.198	.214	79.15	0.300	0.298	0.339	0.317	103.0	4.43	5B	✗	7.1
SwiftBrush [53]†	SD2.1	1	865M	17.20	0.301	0.672	0.458	0.545	27.18	0.314	0.729	0.527	0.612	95.0	3.85	1.4M	✓	4.1
SwiftBrushv2 [7]‡	SD2.1	1	865M	15.98	0.326	**0.782**	0.457	0.577	26.28	0.326	**0.816**	0.543	0.652	139.6	4.91	1.4M	✓	24.1
LCM [44]†	SDXL	1	2.6B	73.75	0.285	0.277	0.254	0.265	82.74	0.285	0.344	0.384	0.363	661.0	13.84	12M	✗	1.3
SDXL-Turbo [64]†	SDXL	1	2.6B	18.98	**0.343**	0.765	0.413	0.536	29.17	**0.343**	0.804	0.518	0.630	180.7	9.24	unk.	✗	unk.
SDXL-Lightning [37]†	SDXL	1	2.57B	20.71	0.331	0.740	0.388	0.509	30.75	0.323	0.760	0.487	0.594	181.2	9.19	30M	✗	unk.
LCM [44]†	SD1.5	4	860M	23.21	0.262	0.666	0.346	0.455	40.37	0.303	0.713	0.460	0.559	592.3	5.88	12M	✗	1.3
LCM-LoRA [45] †	SD1.5	4	860M	26.06	0.323	0.722	0.312	0.436	36.17	0.322	0.768	0.406	0.531	189.9	4.66	12M	✗	1.3
Hyper-SD [57] †	SD1.5	4	860M	21.94	0.326	0.742	0.327	0.454	31.73	0.325	0.804	0.430	0.560	221.9	4.55	unk.	✗	33.3
PCM [73] †	SD1.5	4	860M	21.44	0.316	0.766	0.360	0.490	31.35	0.315	0.770	0.430	0.552	304.3	4.56	3.3M	✗	2
SD-Turbo [64]†	SD2.1	4	865M	16.14	0.335	0.633	0.394	0.468	26,14	0.335	0.694	0.375	0.487	272.2	3.86	unk.	✗	unk.
TCD [86]†	SD2.1	4	865M	18.06	0.319	0.761	0.419	0.540	27.83	0.318	0.795	0.507	0.619	199.2	4.43	5B	✗	7.1
LCM [44]†	SDXL	4	2.6B	17.66	0.327	0.780	0.408	0.536	27.15	0.328	0.810	0.513	0.628	1074.3	13.84	12M	✗	1.3
SDXL-Turbo [64]†	SDXL	4	2.6B	17.79	0.340	0.769	0.431	0.552	27.57	0.341	0.814	0.529	0.641	305.7	9.24	unk.	✗	unk.
SDXL-Lightning [37]†	SDXL	4	2.57B	19.82	0.322	0.715	0.401	0.514	29.32	0.333	0.782	0.457	0.577	310.9	9.19	30M	✗	unk.
Ours	SD2.1	1	865M	**13.09**	0.313	0.634	**0.622**	**0.628**	23.11	0.313	0.697	**0.668**	**0.682**	164.7	4.98	1.4M	✓	2

Table S1. Comparison of our distillation method against other works. Inference Time (ms) and Memory (GB). † indicates that we report results using the provided official code and pretrained models. ‡ denotes that we re-implemented the work and are providing the scores. * indicates that we report results using the provided generated images. "unk." denotes unknown. The best and second-best scores are highlighted in **bold** and underlined, respectively.

$$z_0=\sqrt{\frac{\alpha_0}{\alpha_1}}z_1+\sqrt{\alpha_0}\left(\sqrt{\frac{1}{\alpha_0}-1}-\sqrt{\frac{1}{\alpha_1}-1}\right)\cdot\epsilon^1. \quad (10)$$

With Eq. (7), Eq. (8), Eq. (9), and Eq. (10), the inference is formulated as

$$z_0 = S\epsilon + E_4\epsilon^4 + E_3\epsilon^3 + E_2\epsilon^2 + E_1\epsilon^1. \quad (11)$$

where $S=\sqrt{\frac{\alpha_0}{\alpha_4}}$, $E_1=\sqrt{\alpha_0}\left(\sqrt{\frac{1}{\alpha_0}-1}-\sqrt{\frac{1}{\alpha_1}-1}\right)$, and $E_t=\sqrt{\frac{\alpha_0}{\alpha_t}}\left(\sqrt{\frac{1}{\alpha_{t-1}}-1}-\sqrt{\frac{1}{\alpha_t}-1}\right), t\in[2,4]$. Since ϵ^t can be computed in parallel, we achieve a one-step inference.

This design reduces inference time with one-step sampling, incurs only a minimal increase in memory usage, and achieves significantly improved generation quality.

A.6. Tab. 2 Explanation

Since DrawBench and PartiPrompts are prompt datasets, we used the SD2.1 samples as the "GroundTruth". The FID, Density, and Coverage metrics computed for the SD2.1 model, when calculated with themselves, are 0 and 1, respectively.

B. Additional Results

B.1. Our Additional Samples

Figs. S1 and S2 show our additional samples conditioned on 32 random text prompts.

B.2. Diversity Comparison

In Fig. S3 and Fig. S4, we show additional results for the qualitative comparison of diversity. We observe that both LCM and SD-Turbo tend to generate images with limited diversity, while they generate high-quality results. SwiftBrushv2 is initialized by SD-Turbo, and thus inherits its diversity problem. In contrast, we are able to produce more realistic and diverse results, and close to the ones of SD, indicating our advantage over the baselines.

For the quantitative comparison of diversity in Tab. 2 of the main paper, as shown in Fig. S5, our results are closer to the real images from AFHQ.

Method \ Metrics	Base Model	Step	AFHQ FID↓	AFHQ Density↑	AFHQ Coverage↑	CelebA-HQ FID↓	CelebA-HQ Density↑	CelebA-HQ Coverage↑	DrawBench FID↓	DrawBench Density↑	DrawBench Coverage↑	PartiPrompts FID↓	PartiPrompts Density↑	PartiPrompts Coverage↑	Training Data: Image Free
SD1.5 [58] (*cfg*=7.5)†	–	50	47.16	0.066	0.030	93.94	0.053	0.013	11.95	0.510	0.622	7.36	0.730	0.887	✗
SD2.1 [58] (*cfg*=7.5)†	–	50	51.67	0.053	0.022	89.57	0.018	0.013	0	1	1	0	1	1	✗
InstaFlow [40]†	SD1.5	1	**51.97**	0.058	0.029	131.99	0.026	0.007	25.08	0.223	0.337	17.64	0.457	0.670	✗
LCM [44]†	SD1.5	1	155.63	0.012	0.033	165.74	0.001	0.004	120.98	0.058	0.014	95.65	0.095	0.072	✗
LCM-LoRA [44]†	SD1.5	1	144.15	0.002	0.001	249.82	0.009	0.001	115.63	0.019	0.014	92.52	0.057	0.077	✗
Hyper-SD [57]†	SD1.5	1	72.07	0.064	0.018	126.65	0.055	0.013	26.04	0.397	0.385	17.50	0.677	0.530	✗
SD-Turbo [64]†	SD2.1	1	77.75	**0.142**	0.033	146.22	0.047	0.006	25.75	0.597	0.488	17.40	0.770	0.775	✗
TCD [86]†	SD2.1	1	96.00	0.021	0.009	129.86	0.018	0.002	69.46	0.067	0.081	53.35	0.184	0.264	✗
SwiftBrush [53]†	SD2.1	1	67.60	0.039	0.014	144.03	0.014	0.002	21.48	0.402	0.441	14.43	0.579	0.737	✓
SwiftBrushv2 [7]‡	SD2.1	1	64.99	0.110	0.025	131.89	0.055	0.012	18.57	0.682	0.597	11.32	0.850	**0.865**	✓
LCM [44]†	SDXL	1	106.39	0.022	0.010	211.19	0.004	0.001	83.97	0.043	0.055	64.31	0.130	0.226	✗
SDXL-Turbo [64]†	SDXL	1	77.88	0.004	0.025	261.00	0.002	0.001	32.01	0.602	0.394	17.38	0.791	0.735	✗
SDXL-Lightning [37]†	SDXL	1	76.83	0.053	0.008	131.89	0.078	0.013	28.44	0.351	0.340	18.15	0.606	0.672	✗
LCM [44]†	SD1.5	4	78.00	0.054	0.008	122.44	0.045	0.045	46.23	0.183	0.187	26.84	0.512	0.575	✗
LCM-LoRA [44]†	SD1.5	4	87.54	0.027	0.005	228.50	0.045	0.006	28.45	0.397	0.362	19.75	0.732	0.694	✗
Hyper-SD [57]†	SD1.5	4	62.67	0.132	0.031	**76.16**	0.096	0.020	17.53	0.642	0.569	11.45	0.884	0.853	✗
PCM [73]†	SD1.5	4	60.02	0.108	0.026	86.90	**0.117**	0.016	17.38	0.673	0.581	10.74	**0.887**	**0.865**	✗
SD-Turbo [64]†	SD2.1	4	77.23	0.011	0.005	193.08	0.013	0.001	27.80	0.281	0.371	22.84	0.500	0.648	✗
TCD [86]†	SD2.1	4	54.64	0.063	0.024	103.36	0.078	0.014	**11.46**	0.623	0.431	**7.90**	0.882	0.922	✗
LCM [44]†	SDXL	4	78.72	0.136	0.030	120.19	0.060	0.012	24.97	0.431	0.443	16.27	0.682	0.759	✗
SDXL-Turbo [64]†	SDXL	4	79.00	0.067	0.027	192.97	0.013	0.010	30.05	0.466	0.373	17.12	0.741	0.742	✗
SDXL-Lightning [37]†	SDXL	4	80.23	0.002	0.001	130.76	0.035	0.006	44.52	0.107	0.132	36.12	0.248	0.395	✗
Ours	SD2.1	1	54.48	0.068	**0.071**	116.82	0.116	**0.068**	21.10	**0.685**	**0.616**	16.28	0.852	0.840	✓

Table S2. Quantitative comparison of our distillation method with other approaches based on FID, Density, and Coverage metrics to assess diversity. † indicates that we report results using the provided official code and pretrained models. ‡ denotes that we re-implemented the work and are providing the scores. The best and second-best numbers are marked with **bold** and underlined, respectively.

B.3. Iteration Qualitative Results

For a better demonstration of the iterative process, we present qualitative results at early 6000 steps in Fig. S6. At 1000 iterations, the model already learned meaningful texture information (see Fig. S6 (the fifth row)).

B.4. Comparison with Additional SD-based Models

To demonstrate the effectiveness of our distillation method, we compare it with FasterDiffusion (FasterD) [34], which shares encoder features at certain adjacent time steps and performs the decoder in parallel at these steps. FasterD [34] accelerates 50-step sampling by 1.8x during inference while maintaining generation quality (FID = 12.42 and 22.61 for COCO2014 and COCO2017) (see Tab. S1). We further include the comparison with LCM-LoRA [45][9], PCM [73][10], Hyper-SD [57][11], and TCD [86][12].

Our method achieves one-step inference while preserving quality (FID = 13.09 and 23.11 for COCO2014 and COCO2017) and outperforms LCM-LoRA, PCM, Hyper-SD and TCD across FID, Recall, and F1 evaluation metrics (see Tab. S1), demonstrating its advantage over traditional acceleration methods.

B.5. Comparison with SDXL-based Models

We compare our distillation method against other works based on SD, similar to state-of-the-art methods [7, 53], in the main paper.

To make a full comparison, we further include SDXL-based distillation methods, such as LCM (SDXL) [44][13], SDXL-Turbo [64][14], and SDXL-Lightning [37][15]. Note that the parameter scale of these models exceeds 2.5 billion, comparing with conventional SD-based models which are often below 1 billion. Qualitative and quantitative results are demonstrated in Tabs. S1 and S2, and Fig. S7. As can be observe that, InstaFlow [40], LCM [44] (1-step), and SwiftBrush [53] face challenges in generating high-quality images. LCM [44] (4-step), SD-Turbo [64], SDXL-

[9] https://huggingface.co/latent-consistency/lcm-lora-sdv1-5

[10] https://github.com/G-U-N/Phased-Consistency-Model

[11] https://huggingface.co/ByteDance/Hyper-SD

[12] https://github.com/jabir-zheng/TCD

[13] https://huggingface.co/latent-consistency/lcm-sdxl

[14] https://huggingface.co/stabilityai/sdxl-turbo

[15] https://huggingface.co/ByteDance/SDXL-Lightning

Turbo [64], SDXL-Lightning [37], and SwiftBrushv2 [7] tend to generate results with similar scenes and identities giving the same prompt, leading to the lack of generation diversity. In contrast, our results are closer to the generation quality and diversity of original SD model. Note that, due to the large amount of parameters of the SDXL model, distillting them into 1-step models [37] generally requires 8×80GB GPUs with batch size as 8. We aim to develop more efficient distillation approaches in the future for extremely large T2I diffusion models to reduce the time and space complexity.

C. Limitations, and Future Work

The present study focuses on implementing a loop-free inference with a shared encoder strategy exclusively in image-free distillation. However, we posit that adopting this shared encoder strategy in image-dependent distillation could yield loop-free sampling, thereby enhancing inference speed without compromising on generation quality. This primarily requires engineering efforts.

D. Broader Impacts

TiUE enhances the semantic binding capability in text-to-image synthesis by enhancing text embeddings. However, it also carries potential negative implications. It could be used to generate false or misleading images, thereby spreading misinformation. If *TiUE* is applied to generate images of public figures, it poses a risk of infringing on personal privacy. Additionally, the automatically generated images may also touch upon copyright and intellectual property issues.

E. Ethical Statement

We acknowledge the potential ethical implications of deploying generative models, including issues related to privacy, data misuse, and the propagation of biases. All models used in this paper are publicly available, as well as the base training scripts. We will release the modified codes to reproduce the results of this paper. We also want to point out the potential role of customization approaches in the generation of fake news, and we encourage and support responsible usage of these models. Finally, we think that awareness of open-world forgetting can contribute to safer models in the future, since it encourages a more thorough investigation into the unpredictable changes occurring when adapting models to new data.

F. Reproducibility Statement

To facilitate reproducibility, we will make the entire source code and scripts needed to replicate all results presented in this paper available after the peer review period. We will release the code for the novel color metric we have introduced. We conducted all experiments using publicly accessible datasets. Elaborate details of all experiments have been provided in the Appendices.

Figure S1. Our additional samples.

Figure S2. Our additional samples.

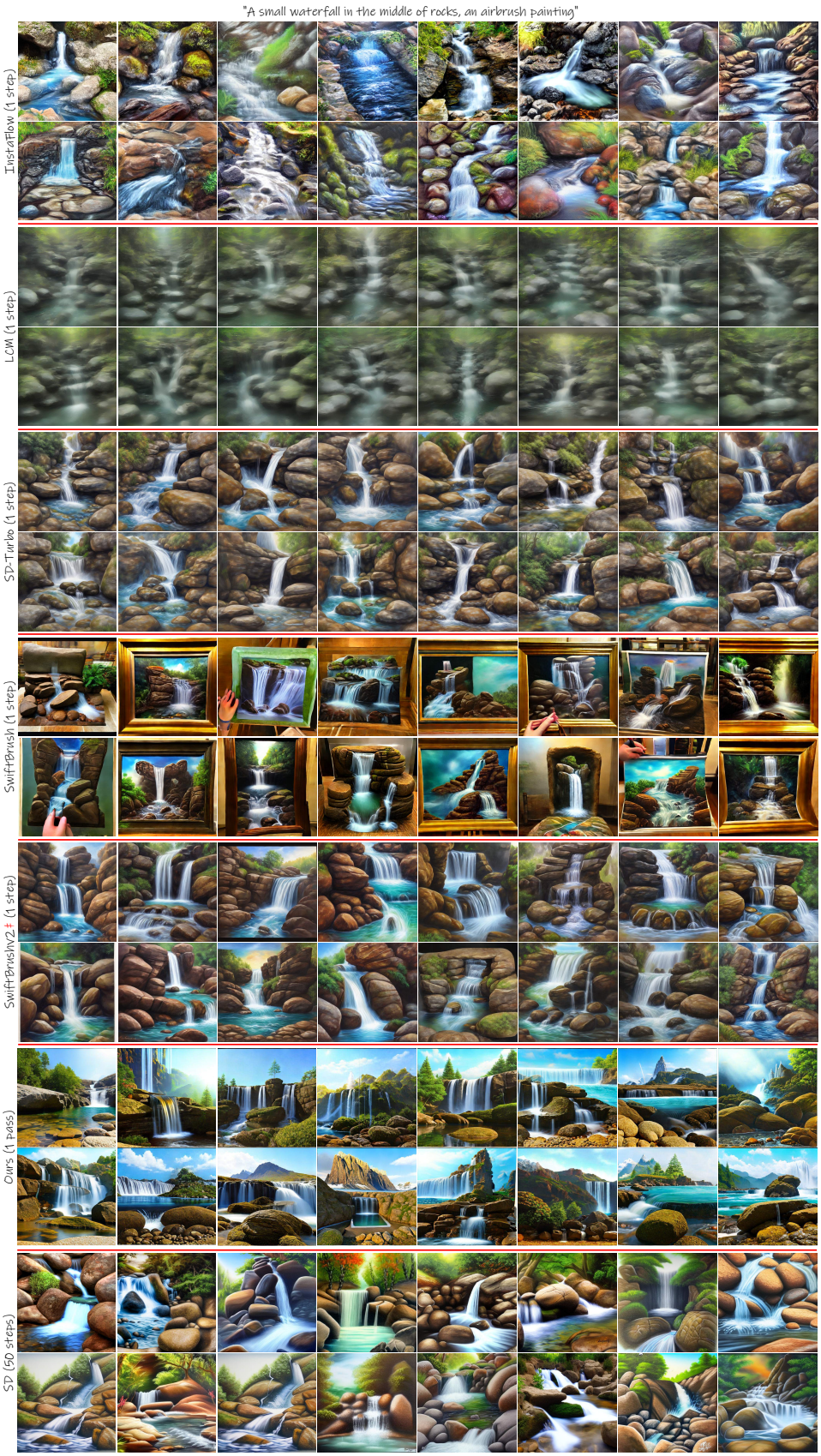

Figure S3. Diversity comparison. Our results are close to the one of SD.

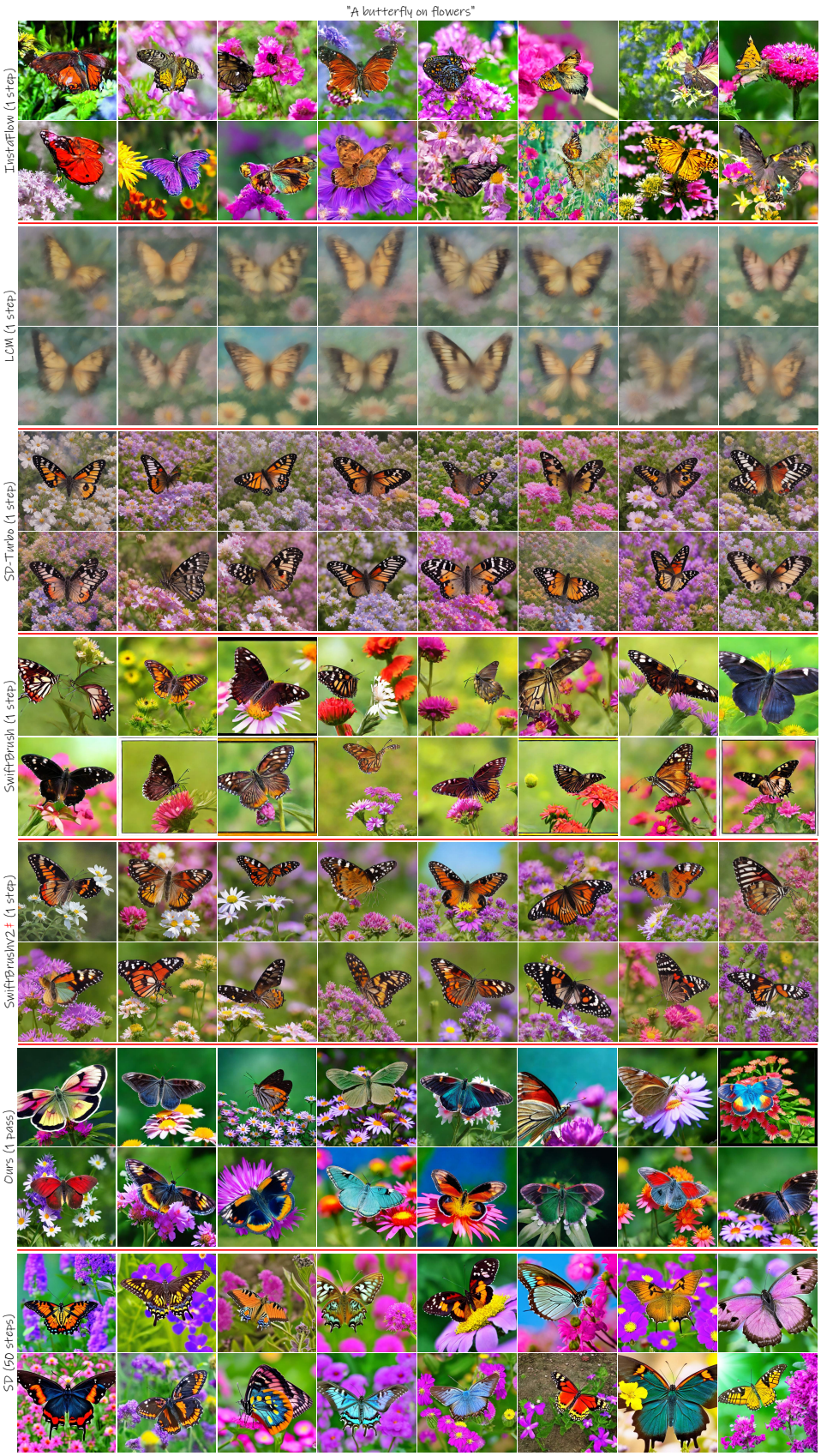

Figure S4. Diversity comparison. Our results are close to the one of SD.

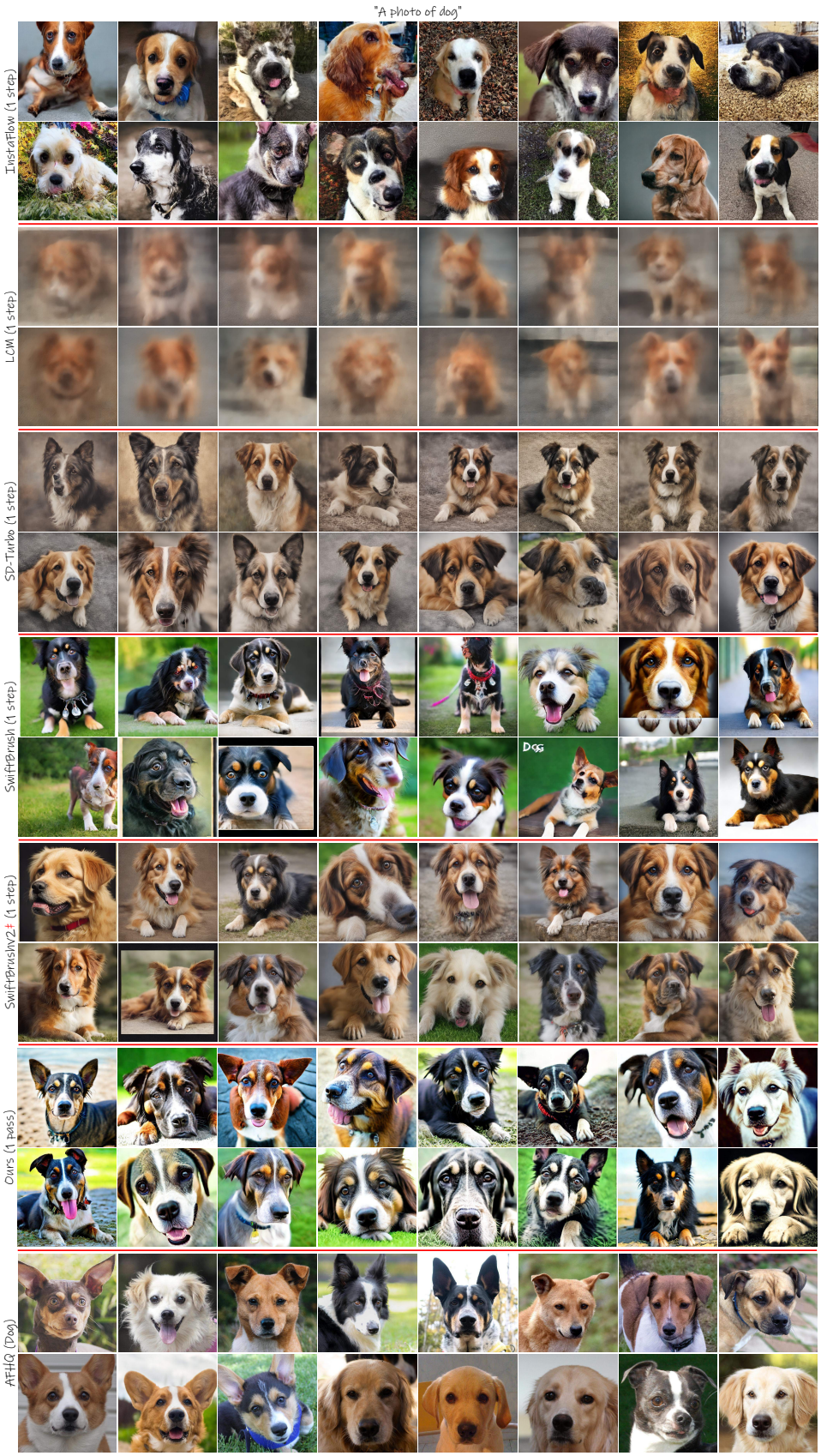

Figure S5. Diversity comparison. Our results are close to the ones of AFHQ.

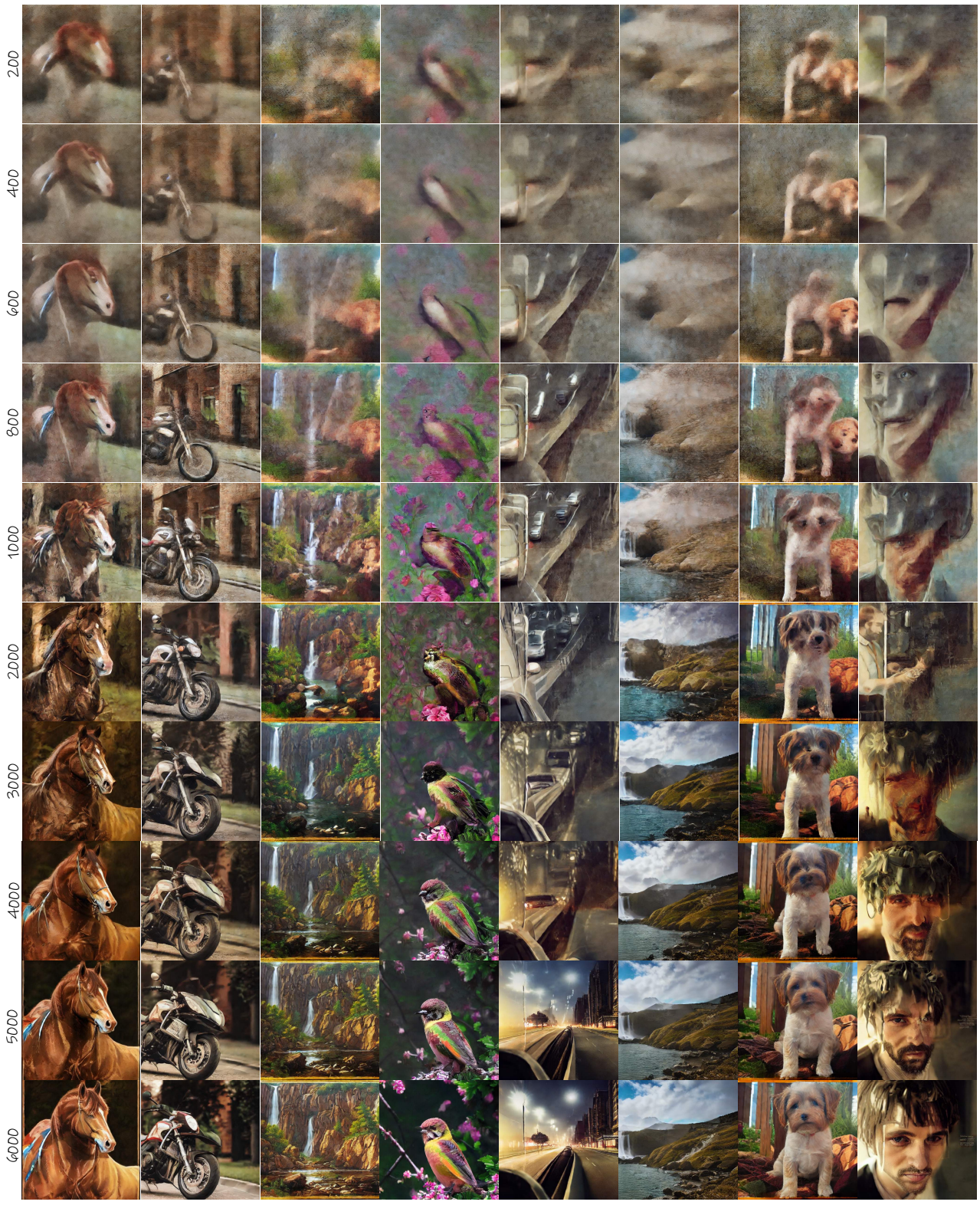

Figure S6. Iteration qualitative results.

Figure S7. Qualitative and diversity comparison. Our results are close to the one of SD.