Training-free Mixed-Resolution Latent Upsampling for Spatially Accelerated Diffusion Transformers

Supplementary Materials

Wongi Jeong*[1] Kyungryeol Lee*[1] Hoigi Seo[1] Se Young Chun[1,2†]
[1]Dept. of Electrical and Computer Engineering, [2]INMC & IPAI
Seoul National University, Republic of Korea
{wg7139, kr.lee, seohoiki3215, sychun}@snu.ac.kr

A. Details on The Derivations

A.1. Derivation of Eq. (7)

Starting from Eq. (4), the conditional distribution of the linear combination of upsampled latent $\mathrm{Up}(\hat{\mathbf{x}}_{e_k})$ and the correlated noise $\mathbf{z} \sim \mathcal{N}(0, \mathbf{\Sigma}')$ is:

$$(a\mathrm{Up}(\hat{\mathbf{x}}_{e_k}) + b\mathbf{z})\,|\mathbf{x}_1 \sim \mathcal{N}\left(ae_k\mathrm{Up}(\mathbf{x}_1), a^2(1-e_k)^2\mathbf{\Sigma} + b^2\mathbf{\Sigma}'\right) \tag{S1}$$

where $\mathbf{\Sigma}' = \mathbf{I} - c\mathbf{\Sigma}$. The conditional distribution of the latent of next stage starting timestep $\mathrm{Up}(\hat{\mathbf{x}}_{s_{k+1}})$ is:

$$\mathrm{Up}(\hat{\mathbf{x}}_{s_{k+1}})|\mathbf{x}_1 \sim \mathcal{N}\left(s_{k+1}\mathrm{Up}(\mathbf{x}_1), (1-s_{k+1})^2\mathbf{\Sigma}\right) \tag{S2}$$

Since we want Eq. (S1) to match Eq. (S2), we can obtain the following equations for the mean and standard deviation, respectively:

$$ae_k = s_{k+1}, \tag{S3}$$

$$a^2(1-e_k)^2\mathbf{\Sigma} + b^2(\mathbf{I} - c\mathbf{\Sigma}) = (1-s_{k+1})^2\mathbf{I} \tag{S4}$$

From Eqs. (S3)–(S4), matching the $\mathbf{\Sigma}$ coefficient to zero gives

$$a^2(1-e_k)^2 = b^2c \implies a(1-e_k) = b\sqrt{c}. \tag{S5}$$

Comparing the $\mathbf{I}$ part,

$$b^2 = (1-s_{k+1})^2 \implies b = 1 - s_{k+1}. \tag{S6}$$

Collecting Eq. (S3), Eq. (S5) and Eq. (S6) yields:

$$s_{k+1} = \frac{e_k}{(1-e_k)/\sqrt{c} + e_k}, \tag{S7}$$

$$a = \frac{1}{(1-e_k)/\sqrt{c} + e_k}, \tag{S8}$$

$$b = \frac{(1-e_k)/\sqrt{c}}{(1-e_k)/\sqrt{c} + e_k}. \tag{S9}$$

By defining the composite term δ_k as

$$\delta_k \equiv (1-e_k)/\sqrt{c}, \tag{S10}$$

Eq. (S7), Eq. (S8), Eq. (S9) can be expressed as:

$$s_{k+1} = \frac{e_k}{\delta_k + e_k}, \quad a = \frac{1}{\delta_k + e_k}, \quad \text{and} \quad b = \frac{\delta_k}{\delta_k + e_k}. \tag{S11}$$

Since $\mathbf{\Sigma}' = \mathbf{I} - c\mathbf{\Sigma} \succeq 0$, $0 \le c \le 1/4$ for 2× nearest-neighbor upsampling.

Two additional facts can be observed here. 1) If $e_k < 1$, then $s_{k+1} < e_k$. This means that adding noise to the upsampled latent always shifts the timestep towards the noise direction. 2) If $e_k = 1$, then $s_{k+1} = 1$, $a = 1$, and $b = 0$. This implies that upsampling a fully denoised latent results in no subsequent stage. However, in this case, the image quality is reduced because there is no remaining step to refine the error caused by the latent upsample.

A.2. Values determined by NT-Matching.

$$\{h_k\}, c = \arg\min_{\{h_k\},c} \mathrm{JSD}(P_{target}(t), P(t)). \tag{S12}$$

We determine the values of $\{h_k\}$ and c in Sec. 4.2 by minimizing the Jensen-Shannon divergence (JSD) between $P_{target}(t)$ (Eq. (11)) and $P(t)$ (Eq. (12)). The resulting values are shown in Tab. S1.

* Authors contributed equally. † Corresponding author.

Table S1. Determined values of Eq. (S12).

Model-acceleration	values we choose		values determined by Eq. (S12)	
	N	e	h	c
FLUX-5×	[4, 9, 15]	[0.3, 0.45, 1.0]	[5.12, 2.64, 2.25]	0.0177
FLUX-7×	[2, 5, 10]	[0.2, 0.3, 1.0]	[8.14, 2.86, 2.19]	0.0255
SD3-2×	[5, 11, 20]	[0.2, 0.3, 1.0]	[6.21, 2.23, 1.97]	0.0586
SD3-3×	[3, 6, 12]	[0.25, 0.3, 1.0]	[6.40, 2.60, 2.23]	0.0255

Table S2. Computational metrics (FLOPs, GPU peak memory) with mixed precision FP16. FLOPs are measured using the `torch.profiler` tool. GPU peak memory is measured using NVIDIA A6000 GPU. We employed FlashAttention [4] for all models excluding ToCa [18].

Method	Accel.	Latency (s)		VRAM usage (GB)
		A100	A6000	
FLUX (50)	-	25.1	49.7	32.8
FLUX (10)	T	5.18	9.99	32.8
Δ-DiT [3]	T	7.42	10.6	45.0
ToCa [18]	T	15.5	29.7	45.0
TeaCache [8]	T	5.23	9.91	34.8
TaylorSeer [9]	T	9.34	17.3	45.0
Bottleneck [15]	S	5.37	10.3	32.8
RALU (Ours)	S	5.04	9.60	32.8
FLUX (7)	T	3.79	7.21	32.8
TeaCache [8]	T	4.21	7.90	34.8
TaylorSeer [9]	T	7.00	12.4	45.0
Bottleneck [15]	S	3.78	7.21	32.8
RALU (Ours)	S	3.75	7.16	32.8

Method	Accel.	Latency (s)		VRAM usage (GB)
		A100	A6000	
SD3 (28)	-	4.04	6.77	15.5
SD3 (14)	T	2.14	3.57	15.5
Δ-DiT [3]	T	2.41	4.04	16.7
ToCa [18]	T	2.62	4.32	17.6
RAS [10]	T	2.07	3.45	20.7
TaylorSeer [9]	T	2.40	4.00	17.9
Bottleneck [15]	S	2.41	3.57	15.5
RALU (Ours)	S	2.04	3.41	15.5
SD3 (9)	T	1.46	2.39	15.5
RAS [10]	T	1.40	2.33	20.7
TaylorSeer [9]	T	1.87	3.11	17.9
Bottleneck [15]	S	1.44	2.40	15.5
RALU (Ours)	S	1.38	2.24	15.5

B. Detailed Experimental Setup

B.1. Experiment compute resources

We generate 1024 × 1024 images using an NVIDIA A100 GPU for all experiments. The latency results reported in Tab. 1 are benchmarked on a single A100 GPU. Furthermore, to demonstrate that RALU is generalizable across different hardware architectures, we also benchmark latency on an NVIDIA A6000 GPU in Tab. S2. In addition, we report GPU memory usage.

B.2. Baseline configurations

B.2.1. Main experiments (Tab.1, 2)

Δ-DiT [3] Δ-DiT is a training-free inference acceleration framework specifically designed for the Diffusion Transformers (DiT) architecture. To address the information loss issues of prior U-Net-based caching, Δ-DiT introduces the Δ-Cache mechanism, which is suitable for DiT's isotropic structure. Δ-Cache selectively caches the deviation (offset) between feature maps instead of the feature maps themselves, thus preserving critical information from the previous sampling step. The core strategy is Stage-Adaptive Acceleration, based on the finding that DiT's front blocks are associated with generating image outlines while the rear blocks handle details. This knowledge is aligned with the diffusion process: Δ-Cache is applied to the rear blocks during the early sampling stages (outline-friendly), and to the front blocks during the later sampling stages (detail-friendly). We set $N = 5$ for FLUX 5× acceleration and $N = 2$ for SD3 2× acceleration, where N is the interval between fully computed steps.

ToCa [18] Token-wise feature Caching (ToCa) is a training-free inference-time acceleration method for Diffusion Transformers that improves efficiency through token-level feature caching. Unlike naïve caching methods that reuse all token features uniformly across timesteps, ToCa selectively caches tokens based on their importance, which is determined by their influence on other tokens (via self-attention), their association with conditioning signals (via cross-attention), their recent cache frequency, and their spatial distribution within the image. ToCa operates by dividing inference into cache periods of length N, where full computation is performed at the first step and cached token features are reused for the next $N - 1$ steps. Within each timestep, a fraction R of the tokens—those deemed less important based on self-attention, cross-attention, cache frequency, and spatial distribution—are selected for caching, while the remaining tokens are recomputed. We set $N = 5$, $N_{\text{total}} = 40$, $R = 90\%$ for FLUX 5× acceleration and $N = 3$, $N_{\text{total}} = 28$, $R = 90\%$ for SD3 2× acceleration, where N_{total} is total inference steps.

TeaCache [8] TeaCache (Timestep Embedding Aware Cache) is a training-free caching approach designed to accelerate diffusion models by selectively reusing intermediate model outputs. It addresses the inflexibility of conventional uniform caching by leveraging the fact that model output differences fluctuate non-uniformly across timesteps. The core idea is to predict output difference using model inputs, which have negligible computational cost. TeaCache uses the timestep-embedding modulated noisy input as the primary indicator for output caching, as this composite input shows a strong correlation with the model output difference. To correct a scaling bias between the estimated input difference and the true output difference, TeaCache employs a simple

polynomial fitting procedure. Caching is determined dynamically using an accumulated relative L1 distance against a threshold (δ), enabling the skipping of redundant computations in consecutive timesteps. We set $N_{\text{total}} = 45$, $\delta = 0.99$ for FLUX 5$\times$ acceleration and $N_{\text{total}} = 30$, $\delta = 0.99$ for FLUX 7$\times$ acceleration, where N_{total} is total inference steps.

RAS [10] Region-Adaptive Sampling (RAS) is a training-free inference-time acceleration method for Diffusion Transformers that dynamically adjusts the sampling ratio for different spatial regions. At each diffusion step, RAS identifies fast-update regions—typically semantically important areas—based on the model's output noise and attention continuity across steps. These regions are refined using the DiT model, while slow-update regions reuse cached noise from the previous step to save computation. To prevent error accumulation in ignored regions, RAS periodically resets all regions through dense steps. Additionally, RAS employs dynamic sampling schedules (e.g., full updates in early steps and gradual reduction thereafter) and key-value caching in attention to maintain quality. RAS dynamically determines which spatial regions require refinement at each step by identifying fast-update areas based on noise deviation and attention continuity. The sampling ratio denotes the proportion of tokens actively updated by the DiT model in each step, while the remaining tokens reuse previously cached noise to reduce computation. We set the sampling ratio to 0.32 for SD3 2$\times$ acceleration and 0.05 for SD3 3$\times$ acceleration.

TaylorSeer [9] Taylorseer introduces a new "cache-then-forecast" paradigm to accelerate DiTs, overcoming the severe quality degradation of prior "cache-then-reuse" methods at high acceleration ratios. It is a training-free approach. The core idea is based on the observation that features in diffusion models evolve along a stable and continuous trajectory across timesteps. Taylorseer leverages Taylor series expansion to predict the features at future timesteps based on cached values from previous steps. The parameter N is the interval (or forced activation period) between fully computed steps. Furthermore, by using an order O greater than 0, the method uses finite difference methods to approximate the features' higher-order derivatives. This enables more accurate modeling of the nonlinear feature trajectory and reduces cumulative prediction errors, which is especially crucial over large intervals (a large N). This predictive strategy allows the model to maintain generation quality even at extreme speedups. We set $N = 7$, $O = 1$, $W = 3$ for FLUX 5$\times$ acceleration, $N = 11$, $O = 1$, $W = 3$ for FLUX 7$\times$ acceleration, $N = 2$, $O = 1$, $W = 3$ for SD3 2$\times$ acceleration, and $N = 4$, $O = 1$, $W = 3$ for SD3 3$\times$ acceleration, where W is the number of fully warm-up steps.

Bottleneck Sampling [15] Bottleneck Sampling is a training-free, inference-time acceleration method that exploits the low-resolution priors of pre-trained diffusion models. It adopts a three-stage high–low–high resolution strategy: starting with high-resolution denoising to establish semantic structure, performing low-resolution denoising in the intermediate steps to reduce computational cost, and restoring full resolution at the final stage to refine details. To ensure stable denoising across stage transitions, Bottleneck Sampling introduces two key techniques: (1) noise reintroduction, which resets the signal-to-noise ratio (SNR) at each resolution change to avoid inconsistencies, and (2) scheduler re-shifting, which adapts the denoising schedule per stage to align with the changed resolution and noise levels. We set the cumulative number of inference steps at the end of each stage $[N_1, N_2, N_3] = [4, 16, 21]$ for FLUX 5$\times$ acceleration, $[3, 11, 15]$ for FLUX 7$\times$ acceleration, $[6, 24, 31]$ for SD3 2$\times$ acceleration, and $[4, 16, 20]$ for SD3 3$\times$ acceleration.

B.2.2. Combining RALU with other accelerations (Tab.3)

To evaluate the feasibility and performance of integrating RALU with various acceleration methods, we established several baselines.

Our primary focus was on combining RALU with temporal acceleration methods, for which we employed Δ-DiT, ToCa, and TaylorSeer. The specific configurations for these baselines were as follows:

- Δ-DiT: The caching interval, denoted as N, was set to $N = 2$.
- ToCa: We configured the caching interval to $N = 3$ and the skip ratio, R (representing the percentage of frames to skip), to $R = 90\%$.
- TaylorSeer: The forced activation period was set to $N = 3$, and the order of the Taylor expansion, O, was set to $O = 1$.

We additionally integrated our method with timestep-distilled models by employing the publicly available weights of FLUX.1-schnell and SD3.5L-Turbo from Hugging Face. For both models, we set the number of function evaluations (NFE) to 4 and configured the cumulative number of inference steps at the end of each stage as $[N_1, N_2, N_3] = [1, 2, 4]$ and $e = [0.3, 0.45, 1.0]$.

B.3. Flow-matching based Diffusion Transformers

For the quantitative comparison, we performed experiments on two flow-matching based diffusion transformers (DiTs) [12]: FLUX-1.dev [1] and Stable Diffusion 3 [12].

FLUX.1-dev FLUX.1-dev is a diffusion-based text-to-image (T2I) synthesis model trained on large-scale data via flow matching, achieving state-of-the-art performance. Despite its high generation quality, the model combines T5-XXL [14] and a CLIP [13] text encoder, resulting in a total of 12 billion parameters. This large model size leads to significant inference latency, posing serious limitations for real-world deployment. In this work, we apply various acceleration methods, including our proposed approach, to FLUX.1-dev and evaluate each method in terms of image quality and

faithfulness to the input text. These evaluations demonstrate the effectiveness of our method.

Stable Diffusion 3 Stable Diffusion 3 (SD3) is a text-to-image synthesis diffusion generative model trained with a rectified flow objective. It conditions on three different text encoders—CLIP-L [13], CLIP-G, and T5-XXL [14]—and has a total of 8 billion parameters. Due to its large model size, SD3 also suffers from non-negligible inference latency, which remains one of the key challenges. In this work, we conduct experiments on SD3 with 2× and 3× speedups to evaluate the effectiveness of our proposed method. In our experiments, we use Stable Diffusion 3 Medium.

B.4. Metrics

ImageReward [17] ImageReward is presented as the first general-purpose text-to-image human preference reward model (RM), designed to effectively learn and encode human preferences for evaluating and improving text-to-image generation models. Its training relies on a systematic annotation pipeline, including rating and ranking, which collected 137k expert comparisons based on real-world user prompts. ImageReward comprehensively captures human preference by evaluating multiple factors, including text-image alignment, image quality, and Harmlessness. ImageReward serves two key roles: as a promising automatic evaluation metric for comparing text-to-image models and individual samples, and as a reward function to directly optimize diffusion models via the Reward Feedback Learning (ReFL) algorithm. In our experiments, we average the ImageReward scores over 5,000 images using the MS-COCO validation set [7].

GenEval [5] GenEval is a comprehensive benchmark designed to evaluate the alignment between generated images and input text prompts in text-to-image (T2I) synthesis. We use two-object, counting, color, and color attribution prompts to evaluate models. It is designed to specifically probe the compositional understanding of T2I models by leveraging existing object detection and other discriminative vision models to verify properties. It can assess how faithfully the generated outputs reflect the semantic content of the given textual descriptions. In our experiments, each prompt is sampled with four different random seeds.

T2I-CompBench [6] T2I-CompBench is a benchmark specifically designed to assess the compositional understanding of T2I generation models. It comprises structured prompts aimed at evaluating a model's ability to accurately associate attributes with corresponding objects, ensuring correct semantic alignment in scenarios involving multiple objects and attributes. For evaluation, we measured performance on spatial, non-spatial and complex sets. By presenting diverse and challenging prompts, T2I-CompBench offers a rigorous evaluation framework for diagnosing issues such as semantic neglect that are prevalent in T2I models. For quantitative evaluation, each prompt is sampled with four different random seeds.

CLIP-IQA [16] The CLIP-IQA metric leverages the pre-trained vision-language model CLIP to assess both quality and abstract perception of images without task-specific training. By using a novel antonym prompt pairing strategy (*e.g.*, "Good photo." vs. "Bad photo.") and removing positional embeddings to accommodate variable input sizes, CLIP-IQA computes the perceptual similarity between images and descriptive prompts. This enables it to evaluate traditional quality attributes like sharpness and brightness as well as abstract attributes such as aesthetic or emotional tone. Extensive evaluations on standard IQA benchmarks and user studies suggest that CLIP-IQA achieves competitive correlation with human perception compared to established no-reference and learning-based methods, while maintaining generality and flexibility. We utilize CLIP-IQA provided by PyIQA [1] with the default prompt setting. We average the CLIP-IQA scores over 5,000 generated images using the MS-COCO validation set [7].

NIQE [11] The Natural Image Quality Evaluator (NIQE) is a no-reference image quality assessment (IQA) metric that operates without any training on human opinion scores or exposure to distorted images. It is a completely blind, opinion-unaware, and distortion-unaware model that measures deviations from statistical regularities observed in natural images. NIQE extracts perceptually relevant natural scene statistics (NSS) features from local image patches and fits them to a multivariate Gaussian (MVG) model built from a corpus of pristine images. The image quality is then quantified as the distance between the MVG model of the test image and that of natural images. Unlike many existing no reference IQA models that are limited to distortion types seen during training, NIQE is general-purpose and performs competitively with state-of-the-art methods, while requiring no supervised learning and maintaining low computational complexity. We utilize NIQE provided by PyIQA [1] with the default prompt setting. We average the NIQE scores over 5,000 images using the MS-COCO validation set [7].

C. Additional Experiments

C.1. Additional ablation studies

Effect of edge detection method. To mitigate aliasing artifacts, we adopt an early upsampling strategy focused on artifact-prone edge regions. In this process, we initially approximate the clean latent ($\hat{x_0}$) using Tweedie's formula, which is then decoded by the VAE to apply Canny edge detection in the image space. A natural question arises: *Is VAE decoding strictly necessary?*

[1] https://github.com/chaofengc/IQA-PyTorch

Table S3. Comparison of different edge detection methods on FLUX.1-dev.

Edge detection methods	TFLOPs ↓	ImageReward ↑	CLIP-IQA ↑	NIQE ↓	T2I-Comp. ↑	GenEval ↑
gradient kernel (5×)	**537.99**	0.996	0.694	6.77	**0.627**	**0.675**
VAE decode → Canny (5×)	540.47	**1.022**	**0.700**	**6.43**	0.626	0.652
gradient kernel (7×)	**423.53**	0.979	**0.685**	6.98	**0.633**	0.662
VAE decode → Canny (7×)	426.01	**0.999**	0.681	**6.87**	**0.633**	**0.682**

Table S4. Comparison of method to determine early upsampling patches on FLUX.1-dev. FLOPs are reported as Mean ± Std.

Early upsampling ratio	TFLOPs ↓	ImageReward ↑	CLIP-IQA ↑	NIQE ↓	T2I-Comp. ↑	GenEval ↑
Adaptive ratio (5×)	550.92 ± 22.91	1.015	0.691	6.56	**0.630**	**0.668**
Fixed r=0.2 (5×)	**540.47** ± 0.00	**1.022**	**0.700**	**6.43**	0.626	0.652
Adaptive ratio (7×)	435.14 ± 12.36	0.986	0.677	**6.81**	**0.634**	0.674
Fixed r=0.2 (7×)	**426.01** ± 0.00	**0.999**	**0.681**	6.87	0.633	**0.682**

Table S5. Comparison of different upsampling methods on FLUX.1-dev, two-stage upsampling framework. We denote the best performance in **bold** and second-best performance with underline.

Upsampling methods	ImageReward ↑	CLIP-IQA ↑
Bilinear	0.9883	0.7045
Bicubic	0.9754	0.7032
Lanczos	0.9848	**0.7077**
Nearest-neighbor (Ours)	**0.9916**	0.7056

Detecting edges directly in the latent space without VAE decoding is inherently inaccurate. High-frequency content in the latent space does not consistently correspond to high-frequency edges in the image space, given the implicit and compressed nature of the information. However, an alternative approach is to utilize a gradient kernel to measure the steepness of change (gradient) between adjacent latent patches and then hypothesize that regions exhibiting a large magnitude of change constitute the edge regions.

Tab. S3 compares FLOPs and performance metrics of different edge detection methods. The gradient kernel method is marginally more efficient in terms of FLOPs as it avoids the VAE decode step. Nevertheless, we observe that the VAE decoding followed by Canny edge detection yields better performance across overall metrics. Despite the slightly higher computational overhead, this method adds a minimal amount of computation (i.e., 2.48 TFLOPs), accounting for less than 1% of the total FLOPs. Therefore, we adopt the VAE decoding and Canny edge detection approach for its higher accuracy in identifying edge regions.

Adaptive upsampling ratio. We select fixed upsampling ratio top-r, where $r = 0.2$ for FLUX-5×, FLUX-7×, SD3-3× settings and $r = 0.3$ for SD3-2× setting. However, recognizing that the proportion of edge patches can vary across images, we also experimented with an adaptive selection strategy based on a fixed edge-strength threshold, rather than a fixed top-r ratio, to determine if this improved image quality. The results in Tab. S4 show that this adaptive approach did not yield improvements in image fidelity although this method introduces computational instability (i.e., high variance in FLOPs) with a slightly higher average computational cost. Given that the fixed-ratio method ensures consistent, predictable computational costs and delivers superior or comparable generation quality, we adopted it for its stability and more favorable performance-efficiency trade-off.

Effect of upsampling method. We employed nearest-neighbor upsampling for feature upsampling. This choice is motivated by the inherent risks associated with conventional image-space interpolation methods when operating in the latent space, where feature correlations are highly complex.

We conducted comparisons with various upsampling methods. Specifically, NT-Matching involves injecting correlated noise, necessitate calculating the exact covariance matrix for the noise. However, this calculation introduces prohibitive computational overhead, rendering it unsuitable for DiT acceleration. Consequently, for a fair comparison in this context, we focused on matching the Signal-to-Noise Ratio (SNR). Tab. S5 shows that employing nearest-neighbor upsampling yields better performance than other upsamplers. Furthermore, the nearest-neighbor approach simplifies the computation of the covariance matrix, which ultimately led to our choice of this upsampling method.

C.2. Visualization of edge detection

As discussed in Tab. S3, applying Canny edge detection after VAE decoding effectively identifies the edge regions. To better understand which areas are actually detected as edge regions, we visualize the detection results in Fig. S1. The figure illustrates the detected edge regions under different early upsampling ratios r.

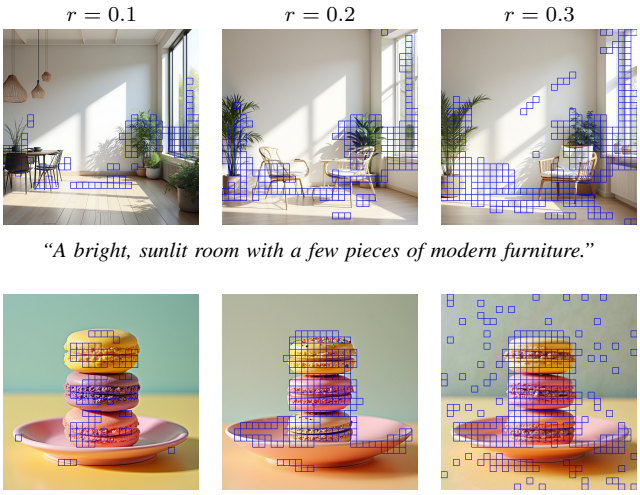

Figure S1. Visualization of detected edge regions under different top-r ratios. The patches identified by the Canny edge detector are highlighted with blue squares. We observe that at $r = 0.3$, the detector effectively captures the majority of structural edges, while also including some non-edge regions. This visualization is conducted under the FLUX-5× acceleration setting.

C.3. Integrating RALU with temporal acceleration

Fig. S2 presents qualitative comparisons obtained by integrating RALU with several temporal acceleration methods. Although the overall speedup increases from 5.53× to as high as 9.03×, the image quality remains well preserved, and no noticeable artifacts are introduced.

C.4. Integrating RALU with timestep-distilled models

Fig. S3 presents qualitative comparisons obtained by integrating RALU with the timestep-distilled models FLUX.1-schnell and SD3.5L-Turbo. Both models operate with an NFE of 4 in combination with RALU, using $[N_1, N_2, N_3] = [1, 2, 4]$ for each. The results show that, even when achieving up to a 15.91× speedup through this integration, the image quality remains largely unaffected and no noticeable artifacts are introduced.

C.5. Additional Qualitative Results

Additional qualitative results are presented in Fig. S4- S9 following the technical appendices. All experimental configurations are provided in the main paper and its appendices. We prepared additional qualitative results following Fig. 6.

C.6. Uncurated Qualitative Results

To demonstrate that our model consistently maintains high generation quality without cherry-picking, we present uncurated qualitative results under 5× and 7× speedup on FLUX. We randomly sampled 96 prompts from the CC12M [2] dataset and generated corresponding images. The results are shown in Fig. S10- S13.

D. Limitations

While region-adaptive early upsampling is broadly applicable to diffusion transformers (DiTs), the Noise-Timestep Matching (NT-Matching) component is tailored specifically for flow-matching–based architectures. Moreover, our method is evaluated only on DiTs; its applicability to other generative backbones, such as diffusion U-Net, remains unverified. Consequently, the generality of NT-Matching beyond flow-matching DiTs has yet to be established.

E. Broader Impact

RALU enables faster and more resource-efficient generation of high-quality images using diffusion transformers, which has the potential to make such models more accessible for real-world or on-device applications. This could democratize creative tools for broader user groups while reducing environmental costs associated with large-scale inference. However, this efficiency gain may also facilitate misuse, such as faster generation of harmful or misleading content. Additionally, the selective focus on visually salient regions (e.g., edges) may implicitly encode or reinforce dataset biases, especially in underrepresented object structures. Care must be taken to evaluate fairness and misuse risks, and we encourage future work to explore responsible deployment strategies alongside technical improvements.

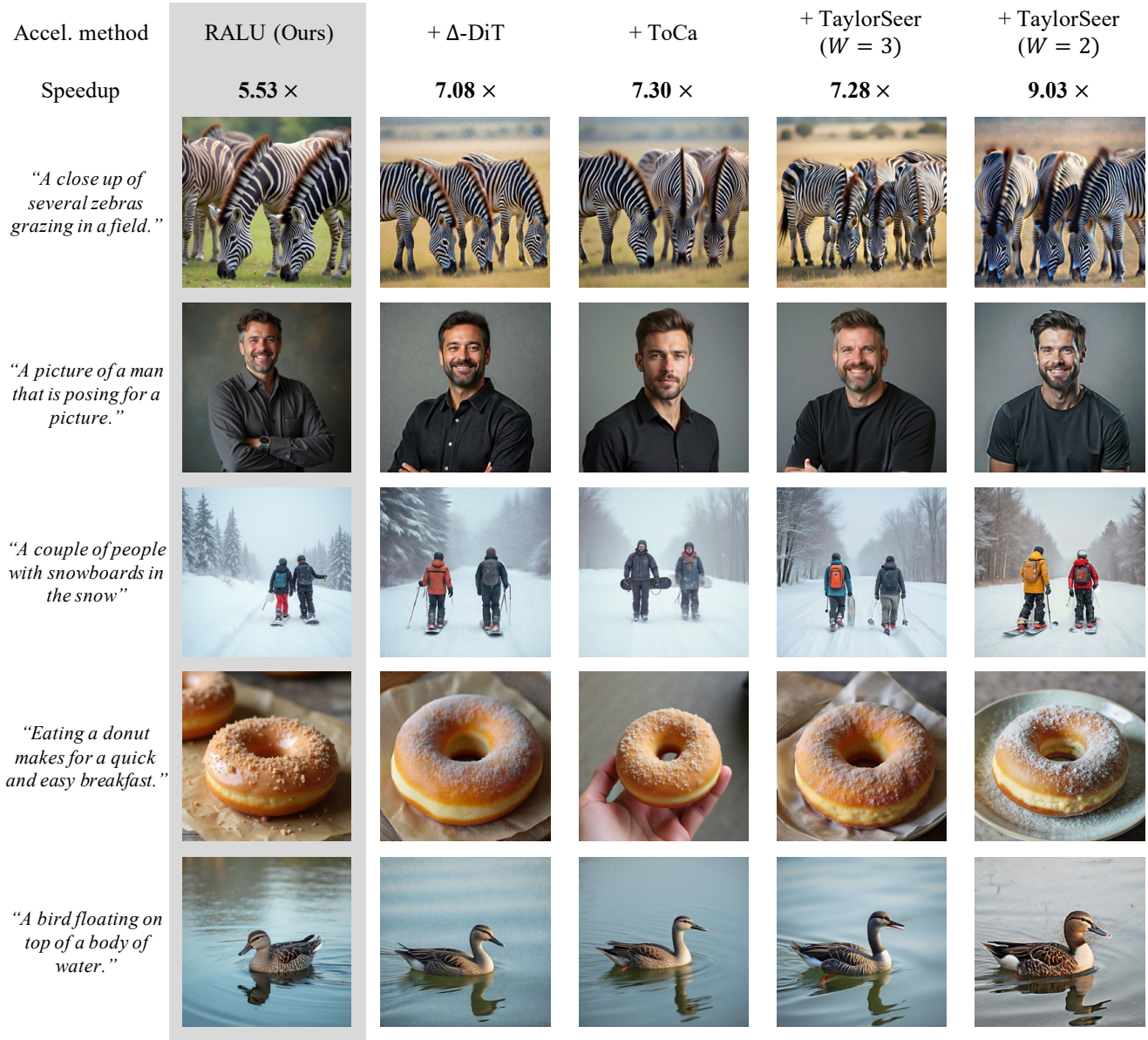

Figure S2. Qualitative comparison of RALU integrated with different temporal acceleration methods.

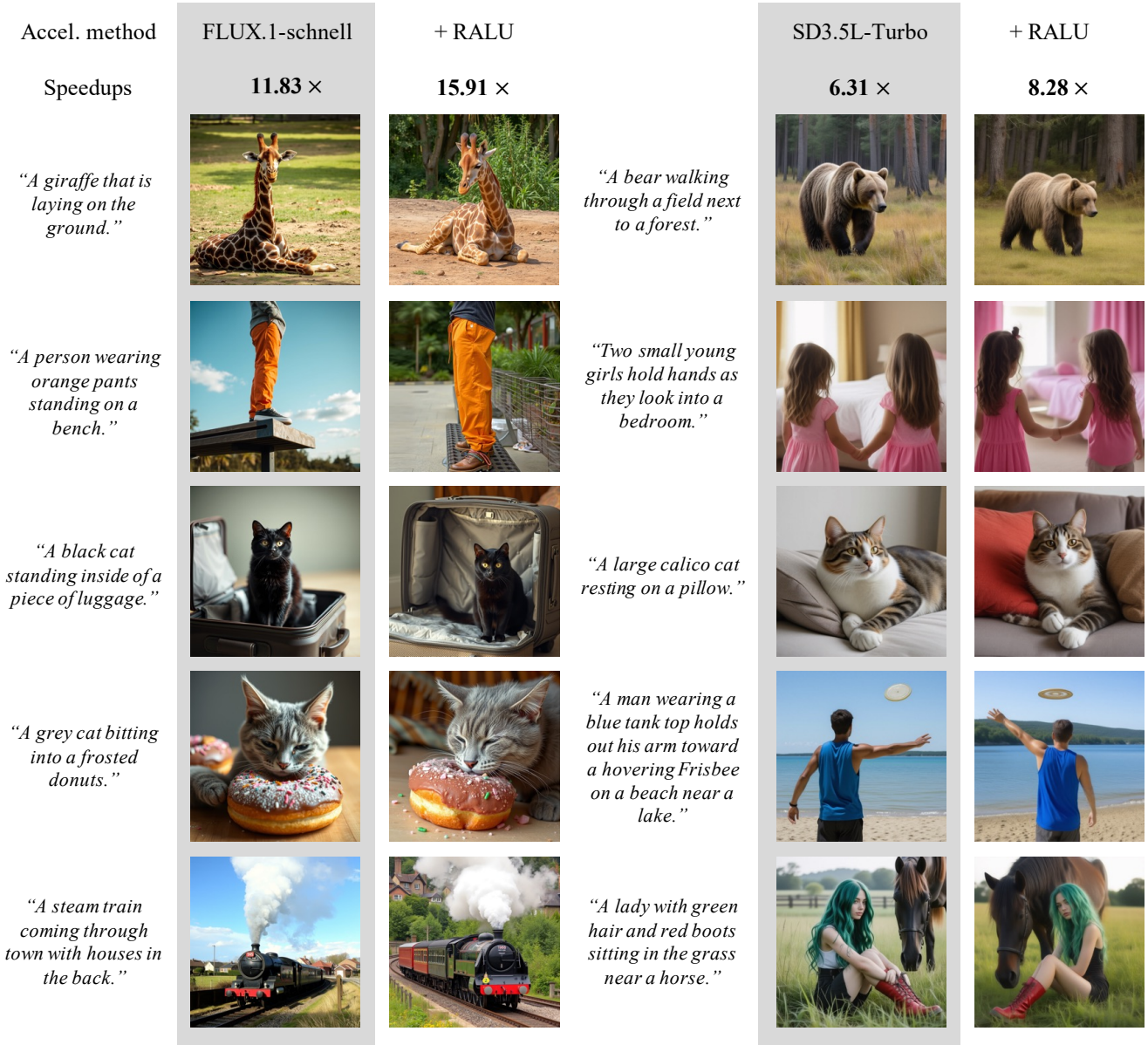

Figure S3. Qualitative comparison of timestep-distilled models integrated with RALU.

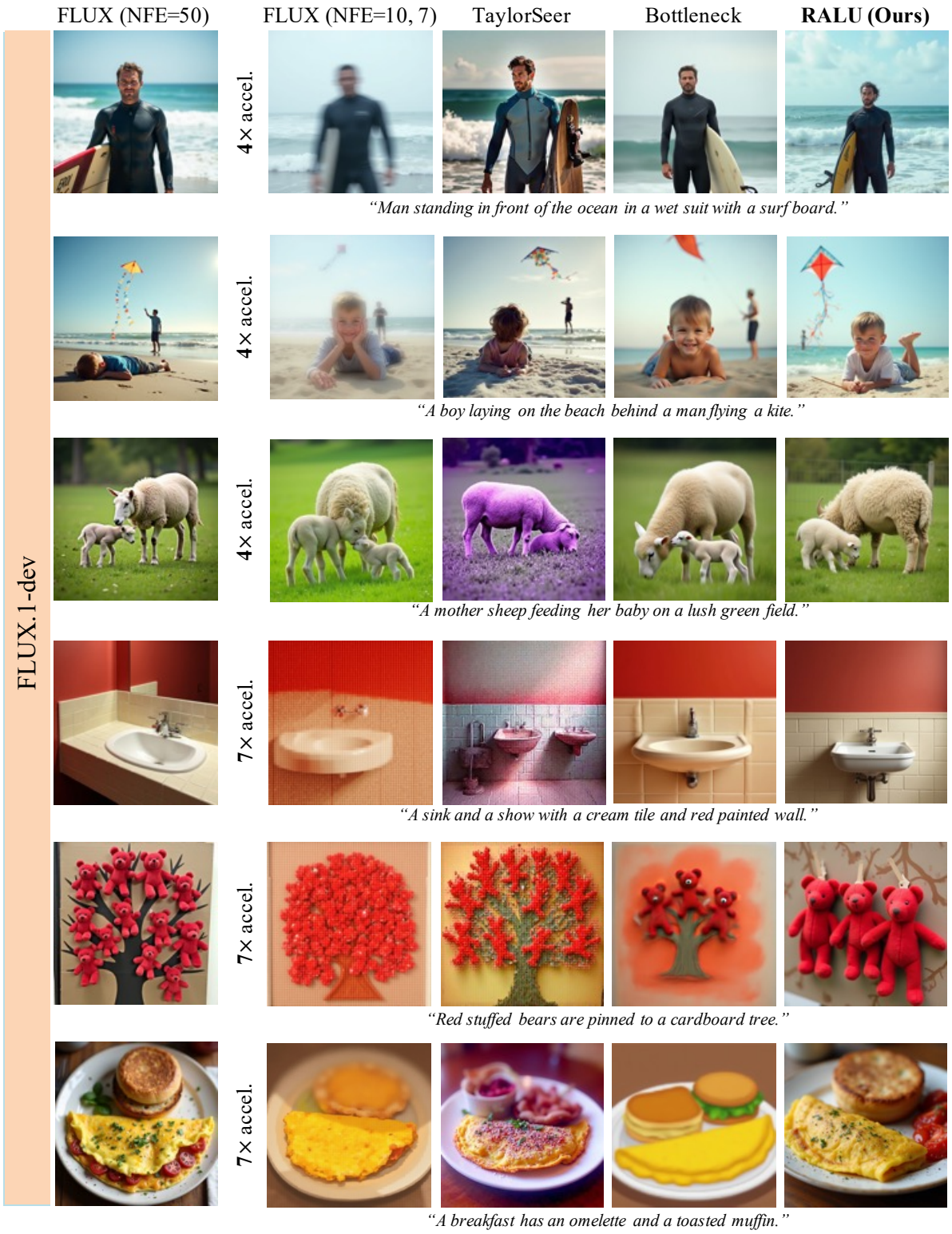

Figure S4. Qualitative comparison of images generated by baseline methods and RALU on FLUX. Best viewed in zoom.

Figure S5. Qualitative comparison of images generated by baseline methods and RALU on FLUX for 5× speedups. Best viewed in zoom.

Figure S6. Qualitative comparison of images generated by baseline methods and RALU on FLUX for 7× speedups. Best viewed in zoom.

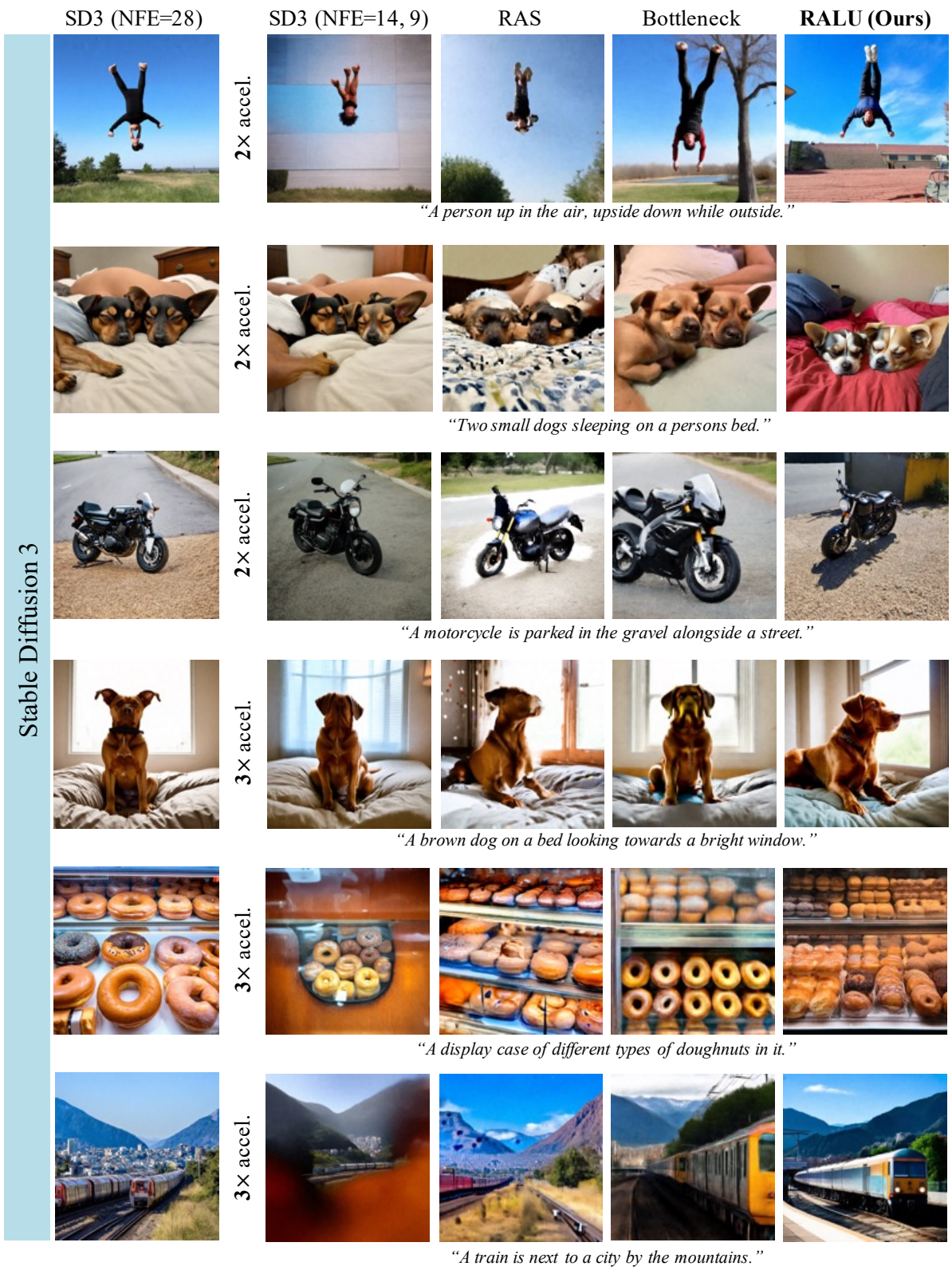

Figure S7. Qualitative comparison of images generated by baseline methods and RALU on SD3. Best viewed in zoom.

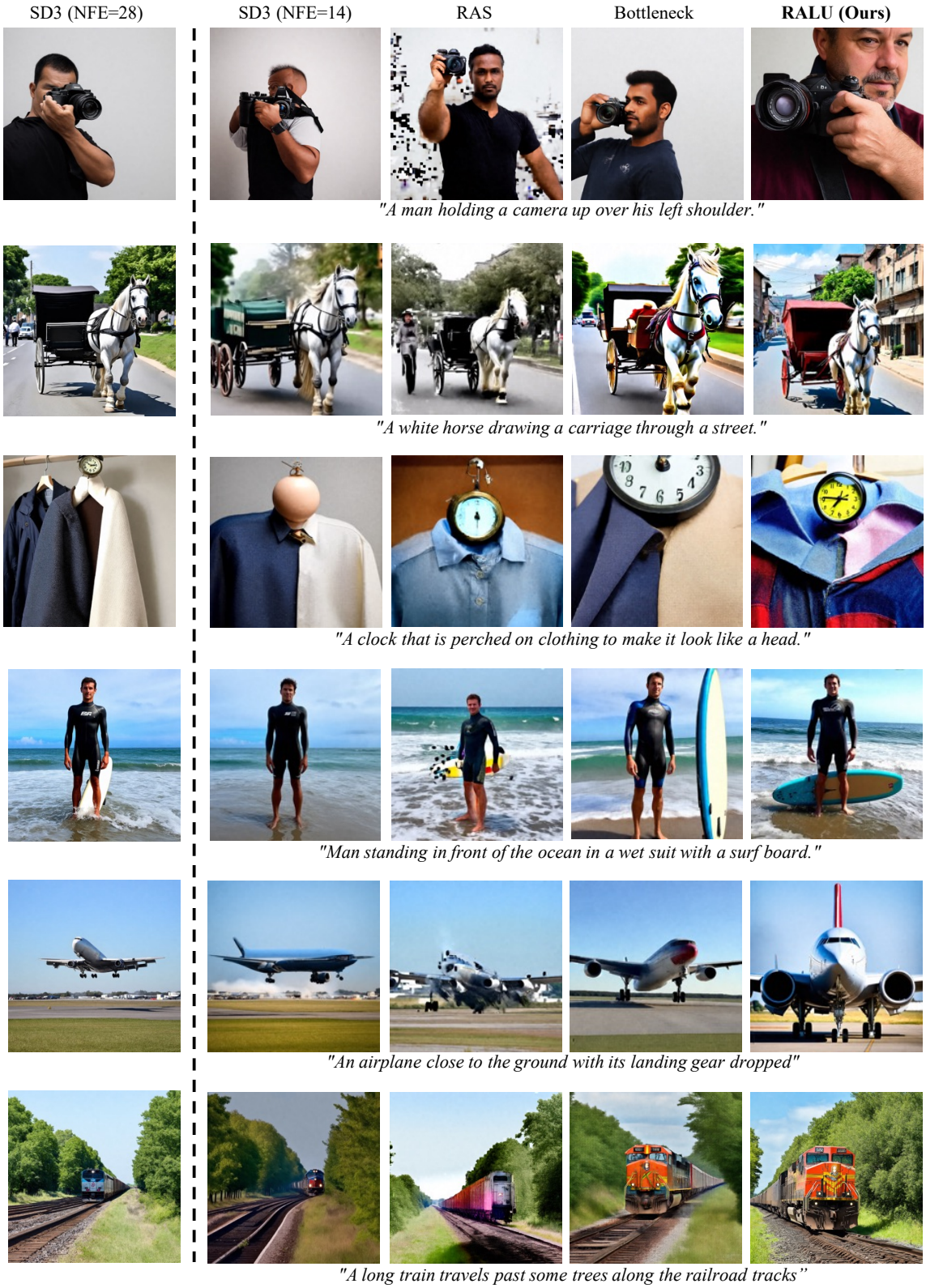

Figure S8. Qualitative comparison of images generated by baseline methods and RALU on SD3 for 2× speedups. Best viewed in zoom.

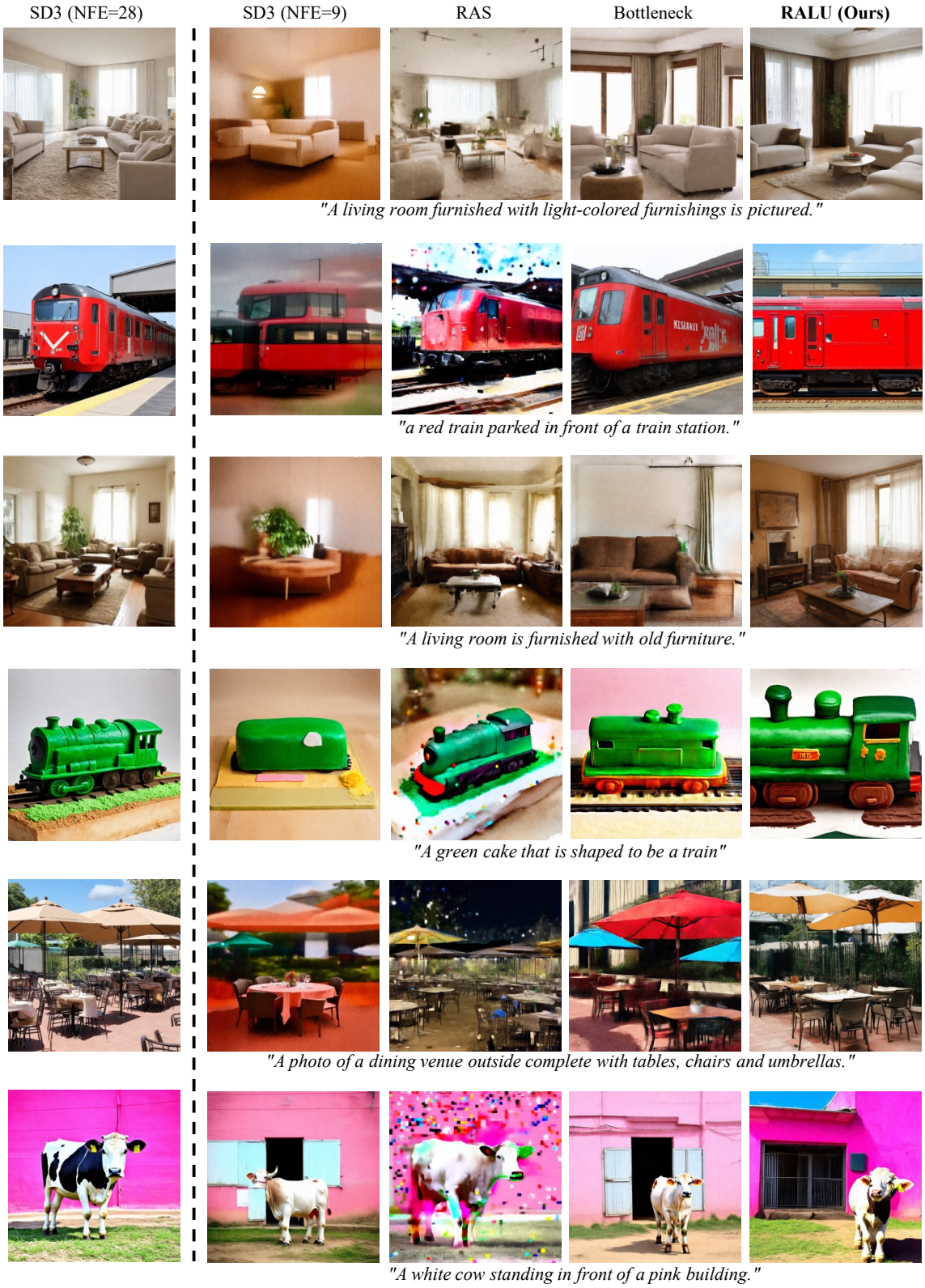

Figure S9. Qualitative comparison of images generated by baseline methods and RALU on SD3 for 3× speedups. Best viewed in zoom.

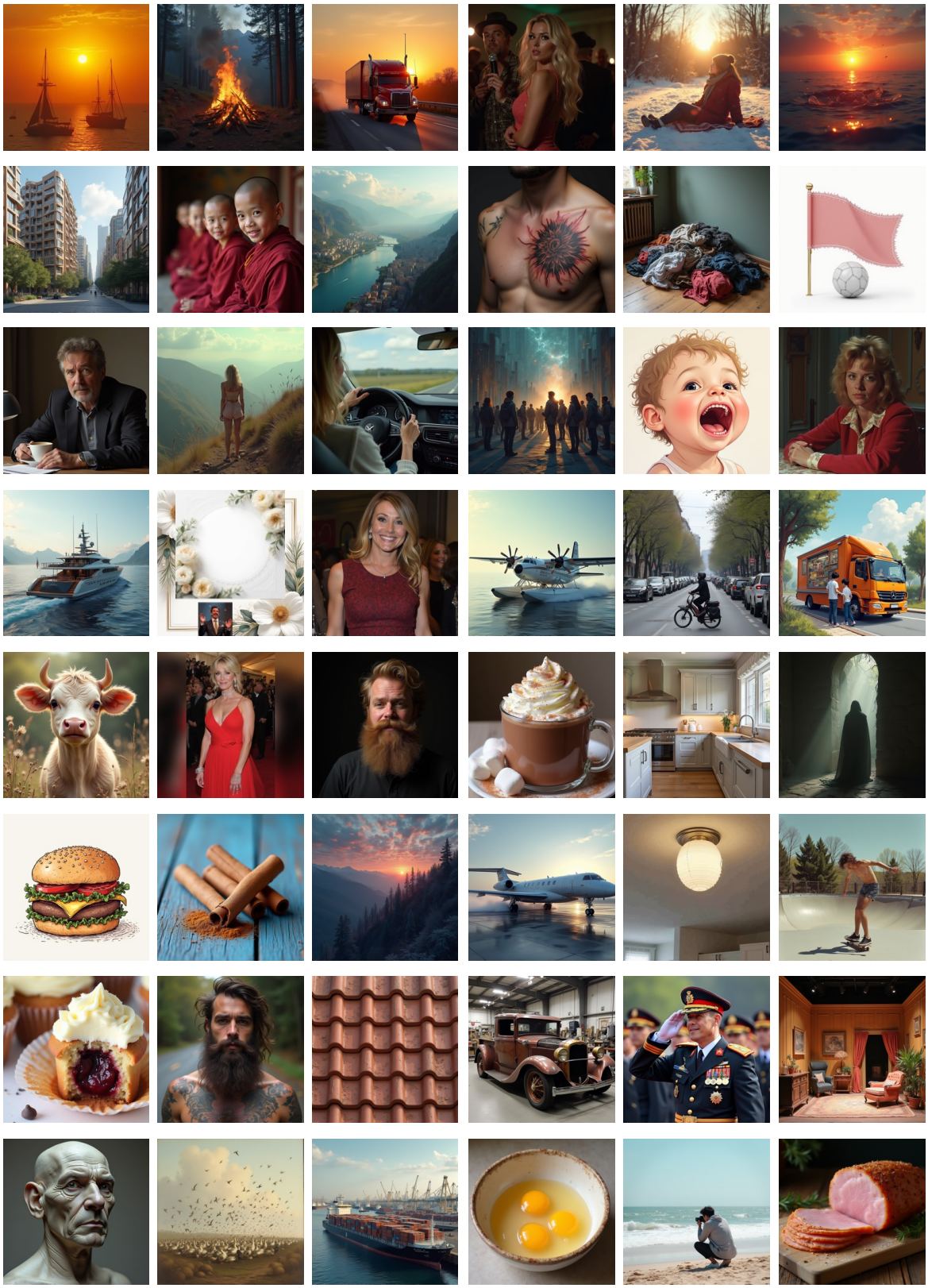

Figure S10. 48 uncurated images generated by RALU on FLUX, 5× speedup.

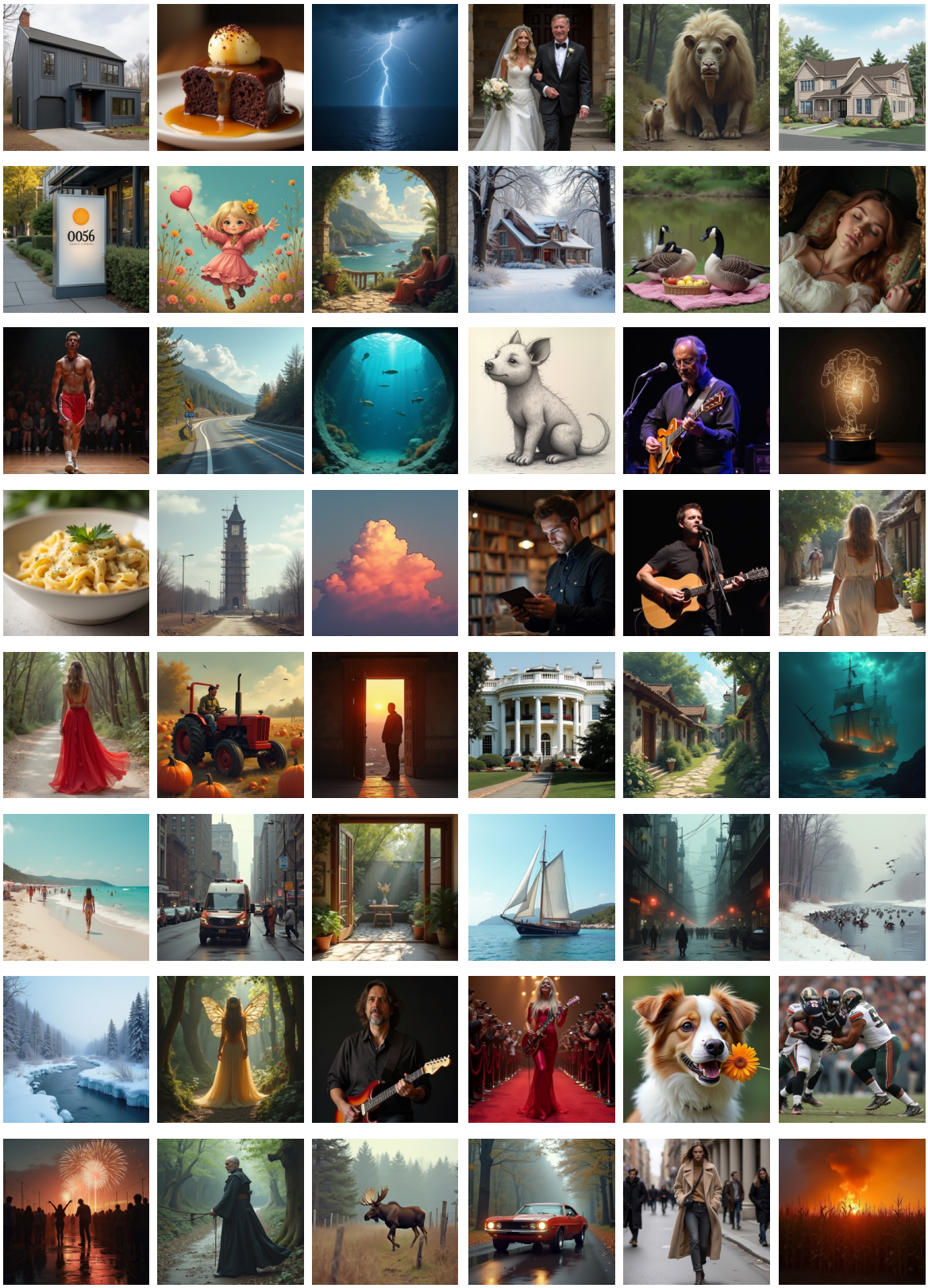

Figure S11. 48 uncurated images generated by RALU on FLUX, 5× speedup.

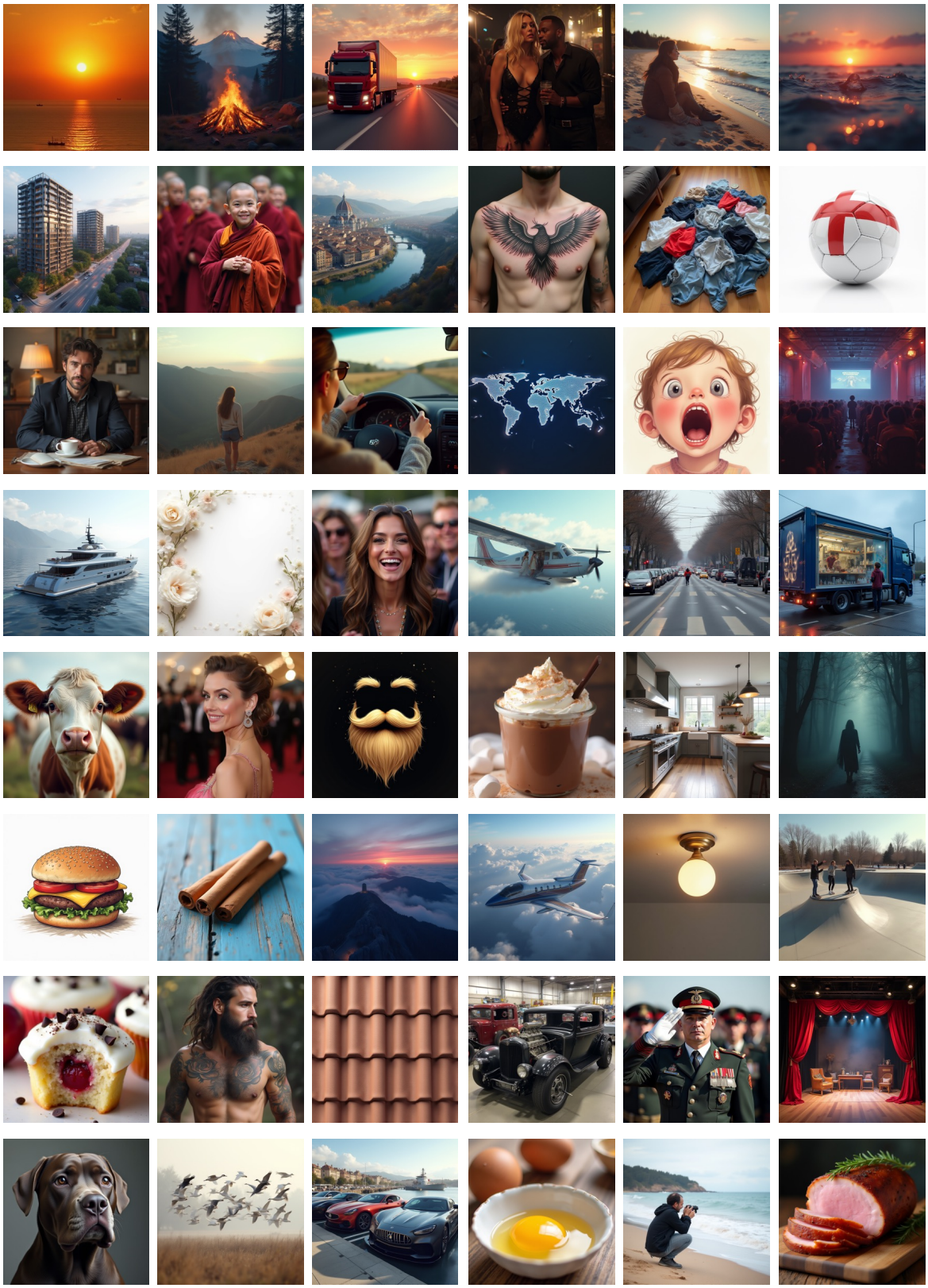

Figure S12. 48 uncurated images generated by RALU on FLUX, 7× speedup.

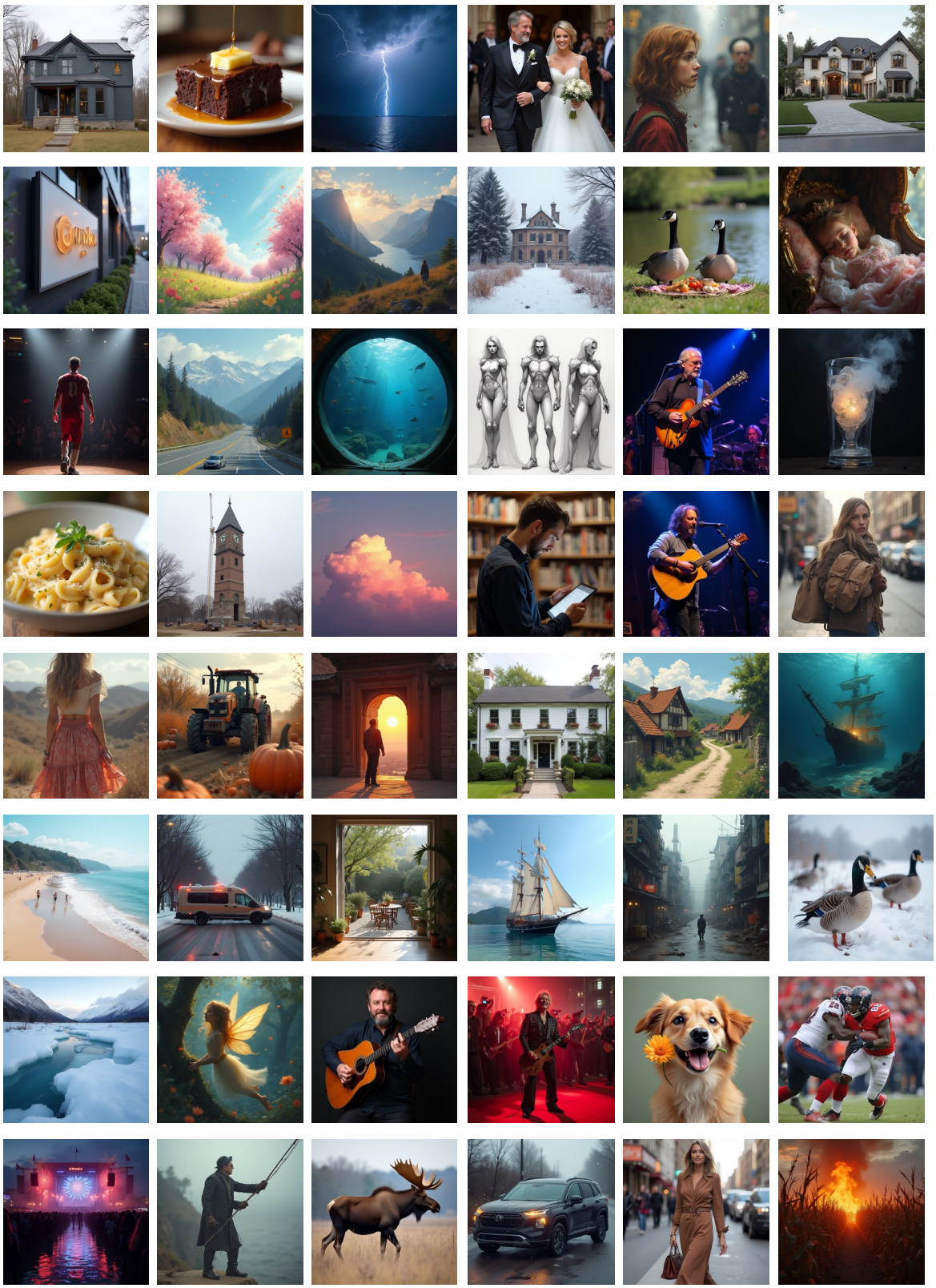

Figure S13. 48 uncurated images generated by RALU on FLUX, 7× speedup.

References

[1] Black Forest Labs. FLUX. https://github.com/black-forest-labs/flux, 2024. 3

[2] Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. *CVPR*, 2021. 6

[3] Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. Δ-DiT: A training-free acceleration method tailored for diffusion transformers. *arXiv:2406.01125*, 2024. 2

[4] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *NeurIPS*, 2022. 2

[5] Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *NeurIPS*, 2023. 4

[6] Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. *NeurIPS*, 2023. 4

[7] 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. *ECCV*, 2014. 4

[8] Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. *CVPR*, 2025. 2

[9] Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers. *ICCV*, 2025. 2, 3

[10] Ziming Liu, Yifan Yang, Chengruidong Zhang, Yiqi Zhang, Lili Qiu, Yang You, and Yuqing Yang. Region-adaptive sampling for diffusion transformers. *arXiv preprint arXiv:2502.10389*, 2025. 2, 3

[11] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal processing letters*, 20(3):209–212, 2012. 4

[12] William Peebles and Saining Xie. Scalable diffusion models with transformers. *CVPR*, 2023. 3

[13] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, et al. Learning transferable visual models from natural language supervision. *ICML*, 2021. 3, 4

[14] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21(140):1–67, 2020. 3, 4

[15] Ye Tian, Xin Xia, Yuxi Ren, Shanchuan Lin, Xing Wang, Xuefeng Xiao, Yunhai Tong, Ling Yang, and Bin Cui. Training-free diffusion acceleration with bottleneck sampling. *arXiv preprint arXiv:2503.18940*, 2025. 2, 3

[16] Jianyi Wang, Kelvin CK Chan, and Chen Change Loy. Exploring clip for assessing the look and feel of images. *AAAI*, 2023. 4

[17] 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*, 2023. 4

[18] Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. *ICLR*, 2025. 2