DC-GEN 2.0: EFFICIENT AND HIGH-FIDELITY IMAGE GENERATION IN DEEPLY COMPRESSED LATENT SPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing diffusion models excel at generating high-quality images, but scaling them to higher resolutions remains computationally expensive. Deep-compression autoencoders reduce latent token counts, yet maintaining reconstruction fidelity and synthesizing fine details in the compressed latent space remain challenging. To address these challenges, we present DC-Gen 2.0, built on our DC-AE 2.0 family. For DC-AE 2.0-Structured, we propose Pyramid-Aligned Decoder Training to facilitate downstream generation while maintaining high reconstruction quality. Building on the structured latent space introduced by DC-AE 1.5, it aligns channel budgets with hierarchical decoder paths and Gaussian-pyramid targets, encouraging coarse-to-fine reconstruction across decoder stages. To improve generation quality under high compression, we introduce Deep-Compression Detail Enhancement, which combines targeted data curation with generation post-training through diffusion adaptation and reinforcement learning. Curated text and face data support fine-detail preservation during autoencoder training and synthesis during diffusion post-training. Experiments show that DC-Gen 2.0 improves reconstruction quality and enables efficient, high-quality image generation under high compression. For FLUX.2-dev, DC-Gen 2.0 f32 achieves 3.39x speedup at 1024 and DC-Gen 2.0 f64 achieves 53.91x speedup at 4096.

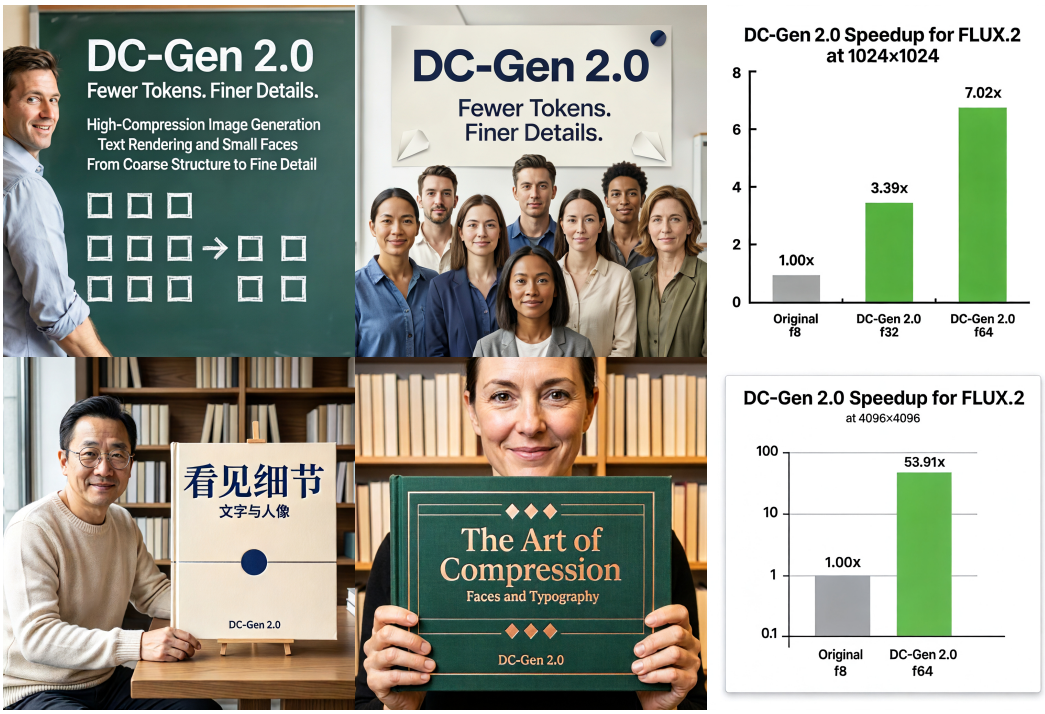

Figure 1: All images, including the speedup charts, are generated by DC-Gen 2.0 f32 at 1024×1024.

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

1 INTRODUCTION

Latent diffusion models are widely used for high-quality image generation (Rombach et al., 2022; Podell et al., 2023; Esser et al., 2024; Black Forest Labs, 2024; Chen et al., 2024a; Xie et al., 2024), but scaling generation to higher resolutions remains computationally expensive. With a fixed autoencoder and patch size, doubling the image height and width quadruples the number of latent tokens, increasing the compute and memory cost of each denoising step (Peebles & Xie, 2023; Chen et al., 2024b). Widely used models, such as Stable Diffusion (Rombach et al., 2022), SD3 (Esser et al., 2024), FLUX.1 (Black Forest Labs, 2024), and FLUX.2-dev (Black Forest Labs, 2025), use autoencoders with a relatively modest spatial compression ratio of $8\times$. Higher spatial compression reduces the number of latent tokens, enabling more efficient generation (Chen et al., 2024b; 2025).

However, high compression can degrade generation quality for two reasons. First, spatial compression can discard fine visual details, reducing autoencoder reconstruction quality. Second, diffusion models can struggle to model fine-grained visual details in deeply compressed latent spaces. These problems are particularly evident in text rendering and small faces, where errors can distort character strokes or facial features. Both must be addressed for high-quality generation under high compression.

To address these challenges, we introduce **DC-Gen 2.0** for efficient, high-quality image generation under high compression. As illustrated in Fig. 2, we address reconstruction and generation limitations at two levels: targeted data and architecture design enhance autoencoder reconstruction, while Deep-Compression Detail Enhancement improves generation quality.

At the autoencoder level, we develop the **DC-AE 2.0** family. Increasing latent channels helps preserve reconstruction quality under high compression, but can make diffusion modeling more difficult (Yao et al., 2025; Chen et al., 2025). Building on the Structured Latent Space introduced by DC-AE 1.5 (Chen et al., 2025), **DC-AE 2.0-Structured** uses **Pyramid-Aligned Decoder Training** to align channel prefixes with hierarchical decoder outputs and Gaussian-pyramid targets (Burt & Adelson, 1983). This design encourages earlier outputs to reconstruct low-frequency structure and later outputs to recover finer details, mitigating the reconstruction degradation associated with structured training. The resulting autoencoder maintains reconstruction quality close to standard DC-AE 2.0, while supporting semantic-to-detail diffusion learning through channel-masked training (Chen et al., 2025).

To improve generation quality under high compression, we introduce **Deep-Compression Detail Enhancement**, combining targeted data curation and generation post-training. The autoencoder uses general-domain images with targeted text and face data to preserve details. A separate corpus of general-domain, English and Chinese text, and small-face images supports diffusion adaptation following DC-Gen (He et al., 2025) and reinforcement learning to improve detail synthesis.

Experiments show that DC-Gen 2.0 improves reconstruction fidelity and generation quality under high compression, while enabling more efficient generation with reduced latent token count.

Our contributions are summarized as follows:

- We present DC-Gen 2.0, a framework that combines deep-compression autoencoding and generation post-training for efficient, high-quality image generation.
- We develop the DC-AE 2.0 family and introduce Pyramid-Aligned Decoder Training, which mitigates reconstruction degradation when building a structured latent space for effective generation.
- We introduce Deep-Compression Detail Enhancement, combining targeted data and generation post-training for faithful detail synthesis at high compression.

2 RELATED WORK

2.1 IMAGE AUTOENCODERS

Image autoencoders map images into latent spaces for generative modeling (Rombach et al., 2022). Stable Diffusion (Rombach et al., 2022), PixArt-α (Chen et al., 2024a), FLUX.1 (Black Forest Labs, 2024), and FLUX.2-dev (Black Forest Labs, 2025) use autoencoders with moderate spatial compression. To reduce latent token counts and accelerate generation, DC-AE (Chen et al., 2024b) develops $32\times$ and $64\times$ autoencoders through residual autoencoding, while Qwen-Image-VAE-2.0 (Zhang et al., 2026) develops $16\times$ and $32\times$ VAEs with expanded channels and semantic alignment.

High spatial compression can degrade reconstruction quality (Chen et al., 2024b; Zhang et al., 2026). More latent channels can alleviate this loss but complicate diffusion training (Yao et al., 2025; Cai et al., 2026). EQ-VAE (Kouzelis et al., 2025) regularizes latent geometry through equivariance. DM-VAE (Ye et al., 2026) matches the aggregate latent distribution to a prior. Spectrum Matching (Ning et al., 2026) aligns latent and image power spectra for frequency-consistent decoding. SSVAE (Liu et al., 2025) uses local-correlation regularization to encourage low-frequency latents.

DC-AE 1.5 (Chen et al., 2025) introduces semantic-to-detail channel ordering and trains diffusion across multiple channel budgets. Our method extends this ordering to decoder supervision by pairing channel prefixes with decoder outputs and matching Gaussian-pyramid targets.

2.2 Text-to-Image Diffusion Models

Recent text-to-image models improve visual quality and prompt following (Esser et al., 2024; Zhao et al., 2026; Tong et al., 2026), while efficient architectures and training strategies reduce cost (Chen et al., 2024a; Xie et al., 2024). SD3 (Esser et al., 2024) and the FLUX family (Black Forest Labs, 2024; 2025) scale transformer-based generation with rectified-flow training. Qwen-Image (Wu et al., 2025) improves bilingual text rendering, while Qwen-Image 2.0 (Zhao et al., 2026) further improves long-text rendering and complex instruction following. Z-Image (Z-Image Team, 2025) targets efficient image generation with a single-stream diffusion transformer. PixArt-α (Chen et al., 2024a) reduces text-to-image training costs, while SANA (Xie et al., 2024) and Seedream 4.0 (Team Seedream et al., 2025) combine deep-compression autoencoders with efficient diffusion transformers.

To avoid the data and compute required to train diffusion models from scratch, DC-Gen (He et al., 2025) adapts pretrained diffusion models to deeply compressed latent spaces through embedding alignment and LoRA fine-tuning. We adopt this approach and improve generation in deeply compressed latent spaces through autoencoder design and Deep-Compression Detail Enhancement.

2.3 Data Curation for Image Generation

Recent work uses targeted training data for autoencoder reconstruction (Zhang et al., 2026) and image generation (Chen et al., 2026a). Qwen-Image-VAE-2.0 (Zhang et al., 2026) supplements general images with text-rich documents and synthetic text rendered on general-domain image backgrounds.

For diffusion training, recent work refines caption content and organization. Seedream 3.0 (Gao et al., 2025) uses specialized captions describing aesthetics, style, and layout, while Qwen-Image 2.0 (Zhao et al., 2026) combines task-specific captioning with synthetic data and high-resolution image filtering. FIBO (Gutflaish et al., 2025) reports better generation with long structured captions than short captions under matched model and image-data settings. A controlled study of text conditioning (Chen et al., 2026b) further finds that enriching captions with image-grounded information reduces diffusion training loss, whereas simply lengthening descriptions gives little improvement. Boogu-Image-0.1 (Chen et al., 2026a) uses targeted synthesis to improve generation of underrepresented Chinese characters. To preserve fine details under high compression, we specifically curate targeted text and face data for autoencoder training and generation post-training.

3 Method

3.1 Overview

DC-Gen 2.0 combines deep-compression autoencoding with generation post-training. We first introduce the autoencoder notation in Sec. 3.2, followed by the DC-AE 2.0 family and Pyramid-Aligned Decoder Training in Sec. 3.3. Sec. 3.4 describes targeted data curation for autoencoder training and generation, followed by the post-training procedure.

3.2 Preliminaries

An image autoencoder consists of an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$. Given an image $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$, the encoder produces a latent $\mathbf{z} = \mathcal{E}(\mathbf{x}) \in \mathbb{R}^{c\times H/f\times W/f}$, and the decoder reconstructs the image as $\hat{\mathbf{x}} = \mathcal{D}(\mathbf{z})$. Here, f is the per-axis spatial compression factor and c is the latent channel dimension. For example, f32c128 denotes $f = 32$ and $c = 128$.

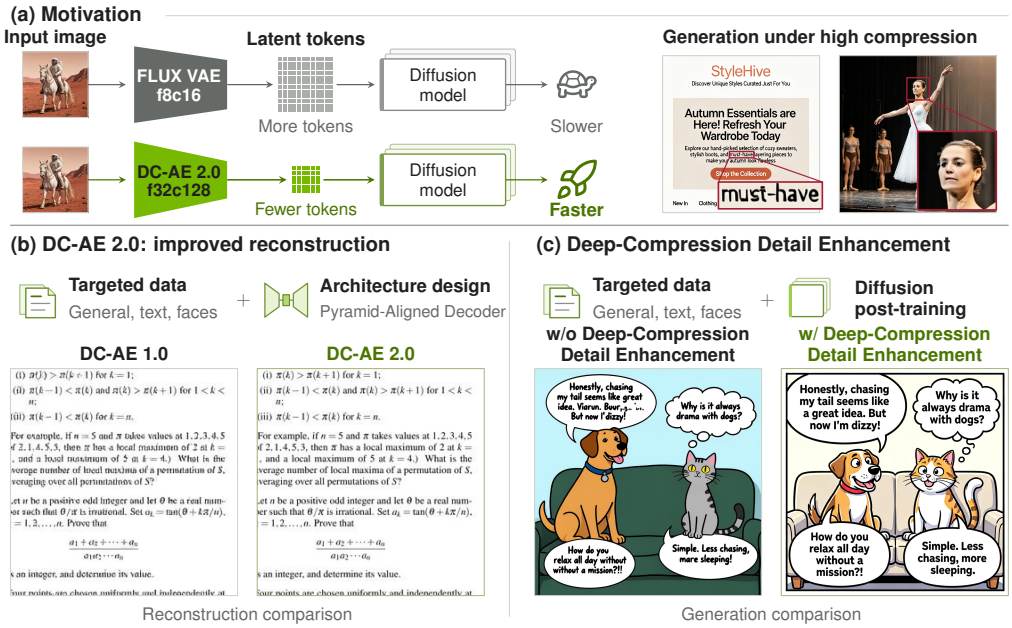

Figure 2: **Motivation and overview of DC-Gen 2.0.** (a) Deep-compression autoencoders reduce latent token counts for efficient generation, but maintaining generation quality remains challenging. (b) Targeted data and architecture design improve autoencoder reconstruction. (c) Deep-Compression Detail Enhancement improves generation quality using targeted text and face data.

3.3 DC-AE 2.0

We develop the **DC-AE 2.0** family. It includes two variants: standard **DC-AE 2.0** and **DC-AE 2.0-Structured**, which builds a Structured Latent Space through Pyramid-Aligned Decoder Training.

Under aggressive spatial compression, increasing the latent channel dimension can improve reconstruction quality but make the resulting latents harder for diffusion models to learn, creating a reconstruction-generation dilemma (Yao et al., 2025). DC-AE 1.5 (Chen et al., 2025) shows that a Structured Latent Space can mitigate this dilemma by progressively organizing the latent channels: early channels capture semantic structure, while later channels add fine visual details. Building on this design, DC-AE 2.0-Structured introduces Pyramid-Aligned Decoder Training to alleviate the reconstruction degradation observed in DC-AE 1.5 (Tab. 2) while retaining its generation benefits. As illustrated in Fig. 3, it aligns ordered channel prefixes with hierarchical decoder outputs and matched Gaussian-pyramid targets. This multi-scale supervision encourages coarse, low-frequency reconstruction at earlier outputs and finer detail recovery at later outputs. Architecture details are provided in Appendix A. Channel-wise frequency analysis is presented in Appendix A.4.

Pyramid-Aligned Decoder Training. Given $\mathbf{z} = \mathcal{E}(\mathbf{x})$, we use ordered channel budgets $\mathcal{C} = \{c_1, \ldots, c_K\}$, where $c_1 < \cdots < c_K = c$, and retain the prefix $\mathbf{z}_{1:c_k}$ under budget c_k. We also construct a K-level Gaussian pyramid (Burt & Adelson, 1983), where $\mathcal{P}_0(\mathbf{x}) = \mathbf{x}$ and each subsequent level applies Gaussian smoothing and $2\times$ spatial downsampling. As shown in Fig. 3, prefix c_k is decoded to its matched hierarchical output and supervised by $\mathcal{P}_{K-k}(\mathbf{x})$. We use $K = 4$ and $c_k = kc/4$: the budgets $c/4$, $c/2$, $3c/4$, and c are supervised by $\mathcal{P}_3$, $\mathcal{P}_2$, $\mathcal{P}_1$, and $\mathcal{P}_0$, respectively, at resolutions from $H/8 \times W/8$ to $H \times W$. For $c = 128$, $\mathcal{C} = \{32, 64, 96, 128\}$.

We denote the decoder path and output head associated with budget c_k by $\mathcal{D}_{c_k}$; all budgets share the decoder backbone. For the channel budgets $\mathcal{B}$ selected in a batch, we optimize

$$\mathcal{L}_{\text{pyr}} = \frac{1}{|\mathcal{B}|} \sum_{c_k \in \mathcal{B}} \mathcal{L}_{\text{rec}} \left(\mathcal{D}_{c_k}(\mathbf{z}_{1:c_k}), \mathcal{P}_{K-k}(\mathbf{x}) \right). \tag{1}$$

Here, $\mathcal{L}_{\text{rec}}$ combines an L1 pixel loss and an LPIPS perceptual loss (Zhang et al., 2018). Each output head upsamples its decoder feature map by $2\times$ to predict an RGB image at the target resolution.

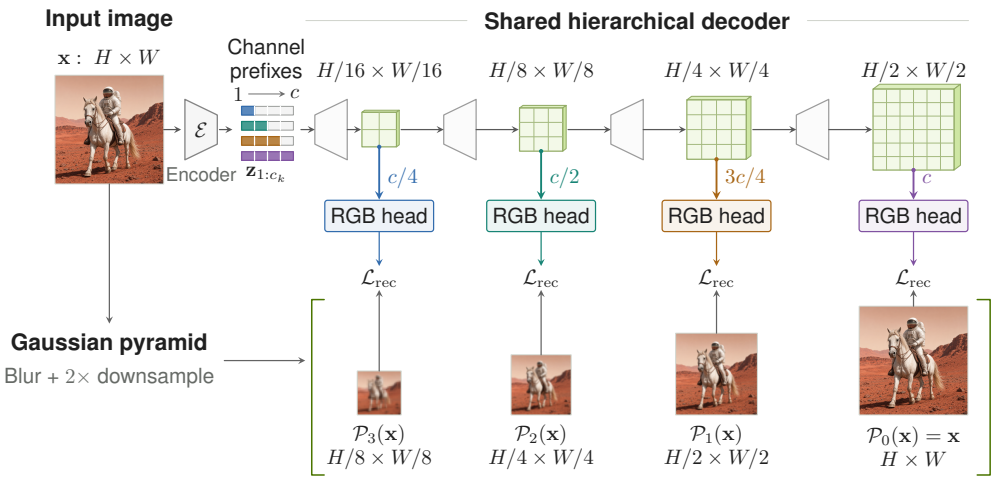

Figure 3: **Pyramid-Aligned Decoder Training.** Channel prefixes of $c/4$, $c/2$, $3c/4$, and c produce progressively finer decoder outputs. Each RGB head upsamples by $2\times$ and is supervised by the corresponding Gaussian-pyramid target derived from the input image.

The autoencoder training curriculum and optimization settings are detailed in Appendix A.3.

3.4 Deep-Compression Detail Enhancement

High compression makes it challenging to preserve fine details during autoencoding and generation. Text rendering and small-face generation are two important cases where detail fidelity affects readability and facial appearance. We therefore complement general-domain images with targeted text and face data for both autoencoder training and generation post-training.

For autoencoder training, we curate a corpus of 145.6M images, comprising 52.0M general-domain images, 42.2M English text-rich images, 24.5M Chinese text-rich images, and 26.9M images containing faces. General-domain data provides broad visual coverage, while the targeted categories supply reconstruction supervision for character strokes and fine facial details that are easily lost under high compression. Both standard and Structured DC-AE 2.0 use this mixture.

For generation post-training, we curate a corpus covering four categories: general-domain images, English text rendering, Chinese text rendering, and small faces, each with 100K images. The text data cover diverse visual settings, including text on everyday objects, document and web layouts, informational and promotional graphics, environmental signage, and text embedded in images. These data support fine-detail synthesis during post-training in the deeply compressed latent space.

Using the curated generation data, we adapt pretrained diffusion models to DC-AE 2.0 following DC-Gen (He et al., 2025), with channel-masked training from DC-AE 1.5 (Chen et al., 2025) for the Structured variant; Appendix B.2 details this adaptation strategy. We then apply RL through joint DMD and image-level reward optimization, building on AnyFlow (Gu et al., 2026) and DMDR (Jiang et al., 2026), to improve generation quality; Appendix B.4 details the training.

4 Experiments

4.1 Evaluation Benchmarks

4.1.1 Autoencoder Benchmarks

Following Qwen-Image-VAE-2.0 (Zhang et al., 2026), we evaluate OmniDoc-TokenBench for text-rich reconstruction and ImageNet (Deng et al., 2009) and FFHQ (Karras et al., 2019) for general reconstruction, using 256×256 for OmniDoc and ImageNet and 1024×1024 for FFHQ. We report PSNR, SSIM, LPIPS, and rFID, together with normalized edit distance (NED) for OmniDoc.

4.1.2 DIFFUSION BENCHMARKS

We evaluate prompt alignment on GenEval (Ghosh et al., 2023) (accuracy) and DPG-Bench (Hu et al., 2024). We report English (EN) and Chinese (ZH) text-rendering quality on LongText-Bench (Geng et al., 2025b), and further use TextPecker (Zhu et al., 2026) to assess glyph structural quality on the same generated images. On images generated from AGHI-QA prompts (Li et al., 2025), we report FaceScore (Liao et al., 2024) mean, 10th percentile (P10), and face detection rate.

4.2 AUTOENCODER EVALUATION

Comparison with State-of-the-Art Autoencoders. Tab. 1 compares autoencoders on OmniDoc-TokenBench, ImageNet, and FFHQ. Models are grouped by total compression, defined as the ratio of RGB image elements to latent elements, $3f^2/c$. At the same total compression ratio of 12×, DC-AE 2.0 f32c256 improves OmniDoc PSNR from 26.24 to 29.78 and NED from 0.9546 to 0.9650 over the FLUX.1 autoencoder (Black Forest Labs, 2024). Fig. 4 shows representative text-rich reconstructions, with enlarged crops for comparing character shapes and stroke boundaries.

Table 1: **Image autoencoder reconstruction on OmniDoc-TokenBench, ImageNet, and FFHQ.** The best result for each metric within each total-compression group is in bold.

Autoencoder	Setting	Total comp.	OmniDoc					ImageNet				FFHQ			
			PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	NED ↑	PSNR ↑	SSIM ↑	LPIPS ↓	rFID ↓	PSNR ↑	SSIM ↑	LPIPS ↓	rFID ↓
FLUX.2-dev	f8c32	6×	**27.72**	**0.9544**	**0.0216**	**0.73**	**0.9535**	**34.33**	**0.9356**	**0.0280**	**0.15**	**40.36**	**0.9675**	**0.0273**	**0.14**
Wan2.1-VAE	f8c16	12×	20.54	0.8282	0.0679	4.57	0.8021	31.29	0.8870	0.0597	0.92	38.16	0.9456	0.0513	0.23
Cosmos-0.1-CI8x8	f8c16	12×	24.29	0.9052	0.0477	2.98	0.9033	32.26	0.9009	0.0649	1.02	39.08	0.9538	0.0636	0.24
Qwen-Image	f8c16	12×	24.94	0.8998	0.0519	4.48	0.9073	**33.42**	**0.9159**	–	–	38.75	0.9512	–	–
Qwen-Image-VAE-2.0	f16c64	12×	26.00	0.9279	0.0382	1.94	0.9244	32.72	0.9086	–	–	39.14	0.9541	–	–
FLUX.1-dev	f8c16	12×	26.24	0.9364	0.0246	**0.55**	0.9546	32.87	0.9155	0.0451	**0.18**	38.14	**0.9573**	0.0421	**0.12**
DC-AE 2.0	f32c256	12×	**29.78**	**0.9602**	**0.0175**	0.73	**0.9650**	33.23	0.9115	**0.0414**	**0.18**	**39.50**	0.9561	**0.0359**	0.13
Wan2.2-VAE	f16c48	16×	21.67	0.8577	0.0525	3.05	0.8310	**31.30**	0.8784	–	–	**37.75**	**0.9386**	–	–
Qwen-Image-VAE-2.0	f32c192	16×	**23.84**	**0.8908**	**0.0497**	**1.98**	**0.8555**	31.13	**0.8785**	–	–	37.52	0.9381	–	–
VAVAE	f16c32	24×	17.50	0.6924	0.0977	4.41	0.3488	27.71	0.7979	0.0977	0.28	32.84	0.8754	0.1331	1.78
LTX-2-VAE	f32c128	24×	18.41	0.7354	0.1192	9.94	0.3569	26.06	0.7925	–	–	33.63	0.9058	–	–
DC-AE 1.5	f32c128	24×	21.56	0.8486	0.0493	2.11	0.7690	29.92	**0.8528**	**0.0638**	0.26	35.67	0.9065	**0.0616**	0.48
Qwen-Image-VAE-2.0	f32c128	24×	22.13	0.8442	0.0642	3.36	0.7065	29.69	0.8423	–	–	35.91	**0.9177**	–	–
DC-AE 2.0	f32c128	24×	**26.66**	**0.9286**	**0.0300**	**1.12**	0.9269	**30.13**	0.8478	0.0706	**0.24**	36.13	**0.9177**	0.0656	0.23
DC-AE 2.0-Structured	f32c128	24×	26.65	0.9280	0.0304	1.19	**0.9287**	**30.13**	0.8451	0.0735	**0.24**	**36.15**	0.9161	0.0676	**0.21**
DC-AE 1.0	f32c32	96×	16.46	0.5937	0.1274	6.40	0.1376	**25.00**	**0.6950**	0.1581	**0.66**	**31.37**	**0.8318**	0.1416	0.54
DC-AE 1.0	f64c128	96×	16.25	0.5854	0.1290	5.62	0.1307	24.74	0.6864	0.1667	1.02	31.36	0.8309	0.1431	0.58
DC-AE 1.5	f64c128	96×	16.75	0.6053	0.1235	7.46	0.1430	24.57	0.6770	**0.1569**	1.20	31.07	0.8232	0.1396	0.57
DC-AE 2.0	f32c32	96×	21.15	0.8189	0.0631	**1.68**	0.6425	24.94	0.6893	0.1585	0.76	31.30	0.8279	0.1400	0.54
DC-AE 2.0	f64c128	96×	21.41	**0.8245**	**0.0626**	1.83	**0.6481**	24.95	0.6882	0.1575	0.70	31.26	0.8270	**0.1395**	0.54
DC-AE 2.0-Structured	f64c128	96×	**21.42**	0.8209	0.0629	1.89	0.6450	24.99	0.6812	0.1644	0.75	31.30	0.8229	0.1448	**0.49**

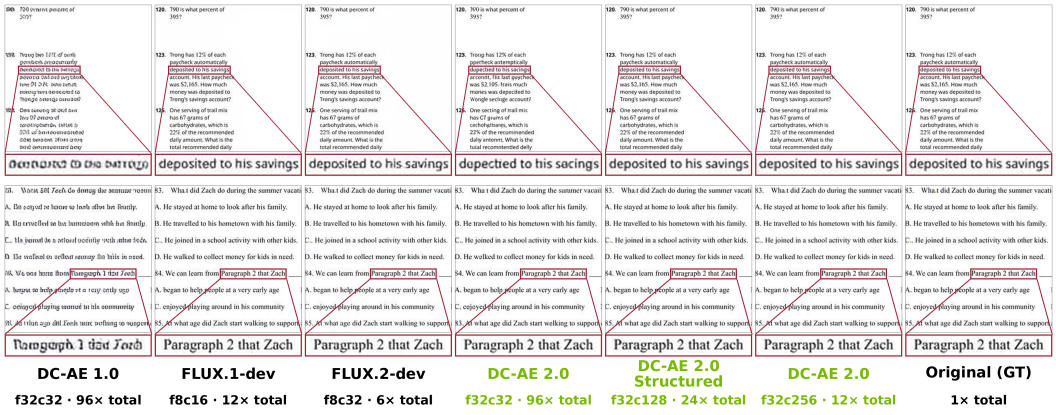

Figure 4: **Qualitative comparison of text-rich image reconstruction.** Reconstructions from representative autoencoders are compared. Autoencoder settings are listed below each column.

Effect of Pyramid-Aligned Decoder Training. Tab. 2 compares standard DC-AE 2.0, the original DC-AE 1.5 structured-training recipe retrained on our data, and DC-AE 2.0-Structured at f32c128. Compared with DC-AE 1.5, DC-AE 2.0-Structured improves all reported reconstruction metrics. These results show that Pyramid-Aligned Decoder Training mitigates the reconstruction degradation of the original structured-training recipe, bringing reconstruction quality close to the standard variant.

Table 2: **Effect of Pyramid-Aligned Decoder Training at f32c128.** DC-AE 1.5 and DC-AE 2.0-Structured use identical initialization, data, and training settings, differing only in whether Pyramid-Aligned Decoder Training is applied. Best results are in bold.

Autoencoder	ImageNet				FFHQ				OmniDoc				
	PSNR ↑	SSIM ↑	LPIPS ↓	rFID ↓	PSNR ↑	SSIM ↑	LPIPS ↓	rFID ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	NED ↑
DC-AE 1.5	29.5052	0.8234	0.0796	0.2502	35.4627	0.9098	0.0711	0.2396	26.2820	0.9131	0.0323	1.2076	0.9252
DC-AE 2.0	30.1292	**0.8478**	**0.0706**	0.2433	36.1277	**0.9177**	**0.0656**	0.2276	**26.6615**	**0.9286**	**0.0300**	**1.1170**	0.9269
DC-AE 2.0-Structured	**30.1323**	0.8451	0.0735	**0.2390**	**36.1490**	0.9161	0.0676	**0.2127**	26.6499	0.9280	0.0304	1.1926	**0.9287**

Effect of Targeted Training Data. Tab. 3 compares standard DC-AE 2.0 f32c128 trained on general-domain images alone with the full mixture, which adds targeted English and Chinese text-rich images and face images. Adding the targeted data improves all five OmniDoc metrics. Fig. 5 illustrates the difference: without these data, character strokes blur or distort.

Table 3: **Effect of targeted data on OmniDoc-TokenBench.** We use standard DC-AE 2.0 f32c128. Targeted data includes English and Chinese text-rich images and face images; the w/o targeted data run uses only general-domain images. Both runs use identical training settings.

Autoencoder Training Data	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	NED ↑
w/o targeted data	24.1228	0.9096	0.0361	1.2525	0.9113
w/ targeted data	**26.6615**	**0.9286**	**0.0300**	**1.1170**	**0.9269**

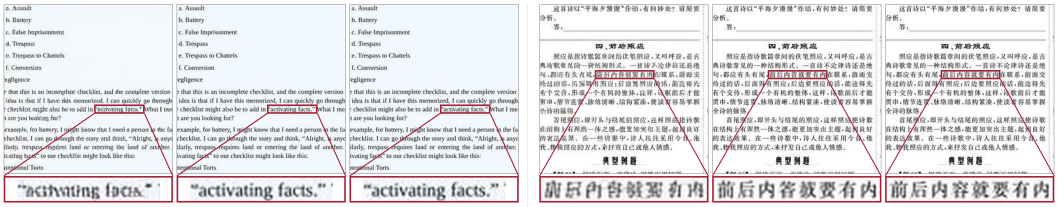

Figure 5: **Qualitative effect of targeted data.** English (left) and Chinese (right) examples compare reconstructions from general-domain-only training and from training with additional text-rich and face images. Red boxes indicate enlarged regions.

4.3 Image Generation Evaluation

Diffusion Configurations. We evaluate image generation with FLUX.1-Krea (Lee et al., 2025) and FLUX.2-dev (Black Forest Labs, 2025) at 1024×1024 using the autoencoder configurations in Tab. 4. These include DC-AE 1.0 f32c32, DC-AE 2.0 f32c32, f32c128, and f64c128, and DC-AE 2.0-Structured f32c128 and f64c128. We additionally evaluate FLUX.2-dev with DC-AE 2.0-Structured f64c128 at 4096×4096. The original models with their native autoencoders serve as references. The training strategy is described in Appendix B.

Overall Generation Results. At 1K, the adapted models achieve comparable or better GenEval and DPG-Bench scores than their respective original models (Tab. 4).

Effect of Structured Latents. For FLUX.2-dev at 1K, Structured latents improve text generation over standard latents. At f32c128, Structured improves LongText-Bench EN and ZH by 0.0017 and 0.0129, respectively; at f64c128, the corresponding gains are 0.0027 and 0.0046. At f64c128, Structured also improves FaceScore mean from 4.3558 to 4.4807 and P10 from 2.0802 to 2.5947, indicating better face quality, including in the lower tail.

Table 4: **Generation quality and efficiency on FLUX.1-Krea and FLUX.2-dev.** Original FLUX.2-dev fails to generate coherent images at 4096×4096. FLUX.1-Krea's text encoder does not support Chinese input. TextPecker evaluates glyph quality on the LongText-Bench images. Throughput is reported for 20 DiT steps with BF16 on one GB300, excluding text encoding and AE decoding.

Autoencoder	AE setting	LongText-Bench		TextPecker		AGHI-QA			Prompt alignment		DiT efficiency	
		EN ↑	ZH ↑	EN ↑	ZH ↑	FaceScore Mean ↑	FaceScore P10 ↑	Detection Rate ↑	GenEval ↑	DPG-Bench ↑	Throughput (img/min) ↑	Speedup ↑
FLUX.1-Krea (1024×1024)												
Original AE	f8c16	0.6171	–	91.32	–	4.394	**2.612**	0.8800	0.6683	83.93	21.36	1.00×
DC-AE 2.0	f32c32	**0.8867**	–	96.73	–	**4.483**	2.557	**0.9800**	0.7255	86.31	74.25	3.48×
DC-AE 2.0-Structured	f32c128	0.8658	–	**97.00**	–	3.652	1.380	0.9700	0.6973	86.18	74.25	3.48×
FLUX.2-dev (1024×1024)												
Original AE	f8c32	0.9506	0.7526	97.23	73.94	4.540	1.478	0.9725	0.8566	87.16	8.95	1.00×
DC-AE 1.0	f32c32	**0.9692**	0.9290	95.30	77.44	4.050	0.129	0.9650	0.8920	87.41	30.30	3.39×
DC-AE 2.0	f32c32	0.9666	0.9290	97.14	82.40	4.459	2.149	0.9700	0.9093	87.33	30.30	3.39×
DC-AE 2.0	f32c128	0.9675	0.9338	**98.23**	87.49	**4.607**	2.605	**0.9925**	0.8938	86.98	30.30	3.39×
DC-AE 2.0-Structured	f32c128	**0.9692**	**0.9467**	98.03	**87.81**	4.545	**2.619**	0.9800	0.8996	87.13	30.30	3.39×
DC-AE 2.0	f64c128	0.9222	0.9187	94.19	76.21	4.356	2.080	0.9900	0.8902	87.26	62.83	7.02×
DC-AE 2.0-Structured	f64c128	0.9249	0.9233	93.56	76.77	4.481	2.595	0.9675	0.8979	87.32	62.83	7.02×
FLUX.2-dev (4096×4096)												
Original AE	f8c32	–	–	–	–	–	–	–	–	–	0.17	1.00×
DC-AE 2.0-Structured	f64c128	**0.9601**	**0.9440**	**98.81**	**92.88**	**4.672**	**2.898**	**0.9775**	0.8781	86.58	8.95	53.91×



Figure 6: **Visual details on LongText-Bench at** 1024×1024**.**

Fine-Grained Generation Quality. LongText-Bench scores do not fully capture glyph quality, as text can remain recognizable despite missing strokes or distorted character shapes. We therefore use TextPecker to assess glyph structural quality. For FLUX.2-dev at 1K, DC-AE 2.0 improves TextPecker over DC-AE 1.0 at f32c32, and increasing the channel dimension to f32c128 yields further gains in both English and Chinese. Standard and Structured f32c128 achieve similar scores. The enlarged text crops in Fig. 6 provide qualitative support, showing more regular strokes and cleaner character contours in the Structured f32c128 examples than in the two f32c32 results.

On AGHI-QA, DC-AE 2.0 outperforms DC-AE 1.0 at f32c32 for FLUX.2-dev, while standard and Structured f32c128 further improve FaceScore mean and P10. The small-face crops in Fig. 7 show clearer eyes and finer skin detail with Structured f32c128 than with either f32c32 model.

Appendix C provides additional English and Chinese text and face comparisons.



Figure 7: **Visual details on AGHI-QA at** 1024×1024**.**

Effect of Diffusion Training Data. Tab. 5 compares training without and with targeted data using FLUX.2-dev and standard DC-AE 2.0 f32c32. With all other training and evaluation settings fixed, using targeted data improves LongText-Bench EN and ZH by 0.2336 and 0.8545, respectively, with a particularly large gain on Chinese. FaceScore P10 also increases from 1.9976 to 2.1493, indicating improved quality in the lower tail of generated faces. These results highlight the importance of training data for text generation and small-face quality under high latent compression.

Table 5: **Effect of training data on generation with a deep-compression autoencoder.** We use FLUX.2-dev with standard DC-AE 2.0 f32c32 and hold the model, training settings, and evaluation protocol fixed, varying only the training data. Best results are in bold.

Training data	LongText-Bench		AGHI-QA		
	EN ↑	ZH ↑	FaceScore Mean ↑	FaceScore P10 ↑	Detection Rate ↑
w/o targeted data	0.7330	0.0745	**4.4896**	1.9976	0.8700
w/ targeted data	**0.9666**	**0.9290**	4.4589	**2.1493**	**0.9700**

Inference Efficiency. At 1K, f32 and f64 reduce FLUX.2-dev image tokens from 4096 to 1024 and 256. Tested on GB300, the 20-step DiT throughput increases from 8.95 to 30.30 and 62.83 images/min, yielding $3.39\times$ and $7.02\times$ speedups, respectively (Tab. 4). Under the same conditions, FLUX.1-Krea improves from 21.36 images/min with its original AE to 74.25 images/min with f32 autoencoders, a $3.48\times$ speedup. At 4K, FLUX.2-dev with Structured f64c128 reduces image tokens from 65,536 to 4096 and increases throughput from 0.17 to 8.95 images/min, a $53.91\times$ speedup. These estimates focus on the diffusion backbone and exclude text encoding and AE decoding.

5 Conclusion

We presented DC-Gen 2.0 for efficient, high-quality image generation in deeply compressed latent spaces. The DC-AE 2.0 family improves reconstruction fidelity. With Pyramid-Aligned Decoder Training, DC-AE 2.0-Structured maintains reconstruction quality while learning a structured latent space that supports semantic-to-detail diffusion learning for improved generation quality. Targeted text and face data further improve the preservation and synthesis of fine visual details through autoencoder training and generation post-training. Experiments demonstrate improvements in reconstruction, text rendering, and small-face generation, together with substantial diffusion speedups at 1K and 4K resolutions. These results highlight the importance of addressing both reconstruction fidelity and diffusion learning to achieve high-quality generation under high compression.

AI USE STATEMENT

Large language models (LLMs) were used solely to assist with manuscript preparation and language refinement, including sentence rephrasing, grammar correction, and improvements to clarity, readability, and flow. All research concepts, ideas, and analyses were independently developed and conducted by the authors. The authors assume full responsibility for the manuscript's content, including any text generated or edited with LLM assistance, and have ensured that such text complies with ethical standards and is free from plagiarism or other forms of scientific misconduct.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 describe the methods, training data, evaluation protocols, and ablations. Appendices A and B provide detailed autoencoder configurations and training recipes for autoencoder training and generation post-training.

ETHICS STATEMENT

Improved text rendering and facial detail may facilitate misleading imagery and impersonation. Training data, including synthetic images, may contain personal information or reproduce social biases. Our evaluations assess visual quality and prompt alignment; deployment requires separate privacy, fairness, and misuse assessments, appropriate safeguards, and compliance with applicable data and model licenses.

REFERENCES

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

Black Forest Labs. FLUX.2: Frontier Visual Intelligence. https://bfl.ai/blog/flux-2, 2025.

Peter J. Burt and Edward H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540, April 1983. doi: 10.1109/TCOM.1983.1095851.

Xin Cai, Zhiyuan You, Zhoutong Zhang, and Tianfan Xue. DA-VAE: Plug-in latent compression for diffusion via detail alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18703–18713, 2026.

Guoxuan Chen, Chufeng Xiao, Haoran Yang, Siyue Xie, Binxiao Huang, Ming Zhang, Cheuk Him Chau, Xinyu Fu, Yingzhao Lian, Tom S. Y. Li, et al. Boogu-Image-0.1: Boosting open agentic multimodal generation via understanding under a minimal budget, 2026a. URL https://arxiv.org/abs/2607.13125.

Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. PixArt-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *International Conference on Learning Representations*, 2024a.

Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. *arXiv preprint arXiv:2410.10733*, 2024b.

Junyu Chen, Dongyun Zou, Wenkun He, Junsong Chen, Enze Xie, Song Han, and Han Cai. DC-AE 1.5: Accelerating diffusion model convergence with structured latent space. *arXiv preprint arXiv:2508.00413*, 2025.

Zilong Chen, Chaorui Deng, Kunchang Li, Hongyi Yuan, and Haoqi Fan. Scaling properties of text conditioning in visual generation, 2026b. URL https://arxiv.org/abs/2607.29679.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. IEEE, 2009.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.

Yu Gao, Lixue Gong, Qiushan Guo, Xiaoxia Hou, Zhichao Lai, Fanshi Li, Liang Li, Xiaochen Lian, Chao Liao, Liyang Liu, et al. Seedream 3.0 technical report, 2025. URL https://arxiv.org/abs/2504.11346.

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

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

Dhruba Ghosh, Hanna Hajishirzi, and Ludwig Schmidt. GenEval: An object-focused framework for evaluating text-to-image alignment, 2023. URL https://arxiv.org/abs/2310.11513.

Yuchao Gu, Guian Fang, Yuxin Jiang, Weijia Mao, Song Han, Han Cai, and Mike Zheng Shou. AnyFlow: Any-step video diffusion model with on-policy flow map distillation. *arXiv preprint arXiv:2605.13724*, 2026.

Eyal Gutflaish, Eliran Kachlon, Hezi Zisman, Tal Hacham, Nimrod Sarid, Alexander Visheratin, Saar Huberman, Gal Davidi, Guy Bukchin, Kfir Goldberg, and Ron Mokady. Generating an image from 1,000 words: Enhancing text-to-image with structured captions, 2025. URL https://arxiv.org/abs/2511.06876.

Wenkun He, Yuchao Gu, Junyu Chen, Dongyun Zou, Yujun Lin, Zhekai Zhang, Haocheng Xi, Muyang Li, Ligeng Zhu, Jincheng Yu, Junsong Chen, Enze Xie, Song Han, and Han Cai. DC-Gen: Post-training diffusion acceleration with deeply compressed latent space. *arXiv preprint arXiv:2509.25180*, 2025. URL https://arxiv.org/abs/2509.25180v1.

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

Dengyang Jiang, Dongyang Liu, Zanyi Wang, Qilong Wu, Liuzhuozheng Li, Hengzhuang Li, Xin Jin, Changsheng Lu, Zhen Li, Mengmeng Wang, Steven Hoi, Peng Gao, and Harry Yang. Distribution matching distillation meets reinforcement learning. In *European Conference on Computer Vision*, pp. 281–299. Springer, 2026. doi: 10.1007/978-3-032-37550-6_16.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Theodoros Kouzelis, Ioannis Kakogeorgiou, Spyros Gidaris, and Nikos Komodakis. EQ-VAE: Equivariance regularized latent space for improved generative image modeling. *arXiv preprint arXiv:2502.09509*, 2025.

Sangwu Lee, Titus Ebbecke, Erwann Millon, Will Beddow, Le Zhuo, Iker García-Ferrero, Liam Esparraguera, Mihai Petrescu, Gian Saß, Gabriel Menezes, and Victor Perez. FLUX.1 Krea [dev]. https://github.com/krea-ai/flux-krea, 2025.

Yunhao Li, Sijing Wu, Wei Sun, Zhichao Zhang, Yucheng Zhu, Zicheng Zhang, Huiyu Duan, Xiongkuo Min, and Guangtao Zhai. AGHI-QA: A subjective-aligned dataset and metric for AI-generated human images. *arXiv preprint arXiv:2504.21308*, 2025.

Zhenyi Liao, Qingsong Xie, Chen Chen, Hannan Lu, and Zhijie Deng. FaceScore: Benchmarking and enhancing face quality in human generation. *arXiv preprint arXiv:2406.17100*, 2024.

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

Shizhan Liu, Xinran Deng, Zhuoyi Yang, Jiayan Teng, Xiaotao Gu, and Jie Tang. Delving into latent spectral biasing of video VAEs for superior diffusability. *arXiv preprint arXiv:2512.05394*, 2025.

Yijun Liu, Jie Huang, Zeyue Xue, Yuming Li, Ruizhe He, Haoran Li, Shijia Ge, and Siming Fu. HPSv3++: Scaling reward models across the full spectrum of diffusion model capabilities. *arXiv preprint arXiv:2606.14657*, 2026. URL https://arxiv.org/abs/2606.14657.

Mang Ning, Mingxiao Li, Le Zhang, Lanmiao Liu, Matthew B. Blaschko, Albert Ali Salah, and Itir Onal Ertugrul. Spectrum matching: a unified perspective for superior diffusability in latent diffusion. *arXiv preprint arXiv:2603.14645*, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

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

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Team Seedream, Yunpeng Chen, Yu Gao, Lixue Gong, Meng Guo, Qiushan Guo, Zhiyao Guo, Xiaoxia Hou, Weilin Huang, Yixuan Huang, et al. Seedream 4.0: Toward next-generation multimodal image generation, 2025. URL https://arxiv.org/abs/2509.20427.

Shengbang Tong, Boyang Zheng, Ziteng Wang, Bingda Tang, Nanye Ma, Ellis Brown, Jihan Yang, Rob Fergus, Yann LeCun, and Saining Xie. Scaling text-to-image diffusion transformers with representation autoencoders, 2026. URL https://arxiv.org/abs/2601.16208.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-Image technical report, 2025. URL https://arxiv.org/abs/2508.02324.

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

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Sen Ye, Jianning Pei, Mengde Xu, Shuyang Gu, Chunyu Wang, Liwei Wang, and Han Hu. Distribution matching variational autoencoder. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2512.07778.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in neural information processing systems*, 37:47455–47487, 2024a.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6613–6623. IEEE, 2024b.

Z-Image Team. Z-Image: An efficient image generation foundation model with single-stream diffusion transformer. *arXiv preprint arXiv:2511.22699*, 2025.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Zekai Zhang, Deqing Li, Kuan Cao, Yujia Wu, Chenfei Wu, Yu Wu, Liang Peng, Hao Meng, Jiahao Li, Jie Zhang, Kaiyuan Gao, Kun Yan, Lihan Jiang, Ningyuan Tang, Shengming Yin, Tianhe Wu, Xiao Xu, Xiaoyue Chen, Yan Shu, Yanran Zhang, Yilei Chen, Yixian Xu, Yuxiang Chen, Zhendong Wang, Zihao Liu, Zikai Zhou, Yiliang Gu, Yi Wang, Xiaoxiao Xu, and Lin Qu. Qwen-Image-VAE-2.0 technical report. *arXiv preprint arXiv:2605.13565*, 2026.

Bing Zhao, Chenfei Wu, Deqing Li, Hao Meng, Jiahao Li, Jie Zhang, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kuan Cao, et al. Qwen-Image-2.0 technical report, 2026. URL https://arxiv.org/abs/2605.10730.

Hanshen Zhu, Yuliang Liu, Xuecheng Wu, An-Lan Wang, Hao Feng, Dingkang Yang, Chao Feng, Can Huang, Jingqun Tang, and Xiang Bai. TextPecker: Rewarding structural anomaly quantification for enhancing visual text rendering. *arXiv preprint arXiv:2602.20903*, 2026. URL https://arxiv.org/abs/2602.20903.

A DC-AE 2.0 Architecture and Training Details

A.1 Backbone and Global Skip Connections

DC-AE 2.0 largely follows the convolutional architecture of DC-AE (Chen et al., 2024b), pairing a tiny encoder with a wider global decoder. The encoder uses global skip connections (GSC) from Qwen-Image-VAE-2.0 (Zhang et al., 2026): each shortcut spans a residual-block stack and its subsequent downsampling operation. The decoder mirrors this topology, with a shortcut spanning upsampling and the subsequent residual blocks. This symmetry concerns stage topology, not channel widths.

Encoder shortcut. Let $\mathbf{h} \in \mathbb{R}^{C_{\text{in}} \times h \times w}$ be the input to a resolution-changing stage, and let F_{enc} denote its residual-block stack. The stage produces

$$\mathbf{h}' = \text{Down}_2(F_{\text{enc}}(\mathbf{h})) + \text{Avg}_g(\text{PU}_2(\mathbf{h})), \qquad g = \frac{4C_{\text{in}}}{C_{\text{out}}}, \tag{2}$$

Here PU_2 rearranges each 2×2 spatial neighborhood into channels. The shortcut groups the $4C_{\text{in}}$ channels into C_{out} groups of size g and averages each group. Its output has shape $C_{\text{out}} \times h/2 \times w/2$. The learned branch implements Down_2 as a 3×3 convolution producing $C_{\text{out}}/4$ channels, followed by pixel unshuffle. Channel widths make both divisions integral.

Decoder shortcut. For an upsampling stage, the learned branch applies upsampling before the residual-block stack F_{dec}:

$$\mathbf{h}' = F_{\text{dec}}(\text{Up}_2(\mathbf{h})) + \text{PS}_2(\text{Repeat}_r(\mathbf{h})), \qquad r = \frac{4C_{\text{out}}}{C_{\text{in}}}. \tag{3}$$

Here, Repeat_r repeats each channel r times, and pixel shuffle PS_2 rearranges the resulting $4C_{\text{out}}$ channels into a feature map of shape $C_{\text{out}} \times 2h \times 2w$. The learned Up_2 uses a 3×3 convolution producing $4C_{\text{out}}$ channels, followed by pixel shuffle.

Both shortcuts are parameter-free and are added directly to their learned branches. Each spans one complete resolution-changing stage, rather than connecting encoder features to the decoder. We disable the separate shortcut inside the learned resampling operator when using GSC, while retaining the identity residual connections within individual blocks. The deepest stage, which preserves spatial resolution, uses residual blocks without an additional stage-level shortcut.

A.2 Shared Architecture Configurations

All configurations use the tiny encoder and global decoder in Tab. 6. The f32 models share stage widths and depths, changing only c and the latent projections. For f64, an additional encoder downsampling stage is appended at the deepest end, from $H/32 \times W/32$ to $H/64 \times W/64$, with width 1024. The global decoder adds the corresponding deepest upsampling stage. The latent projections follow the deepest encoder stage and precede the deepest decoder stage, using 3×3 convolutions to map $1024 \to c$ and $c \to 1024$ channels. Both variants share this backbone.

Table 6: Shared tiny encoder and decoder configuration. The encoder runs left to right and the decoder right to left. The last column is used only for f64; depth counts residual blocks.

Spatial size	$\frac{H}{2} \times \frac{W}{2}$	$\frac{H}{4} \times \frac{W}{4}$	$\frac{H}{8} \times \frac{W}{8}$	$\frac{H}{16} \times \frac{W}{16}$	$\frac{H}{32} \times \frac{W}{32}$	$\frac{H}{64} \times \frac{W}{64}$
Encoder width	64	128	512	1024	1024	1024
Encoder depth	5	10	4	4	4	4
Decoder width	256	512	512	1024	1024	1024
Decoder depth	5	10	4	4	4	4

Each residual block contains two 3×3 convolutions with no channel expansion, SiLU activation after the first convolution, and channel-wise RMS normalization after the second. This configuration contains no attention blocks. The stem is a 3×3 convolution from 3 to 16 channels followed by pixel unshuffle, yielding 64 channels at $H/2 \times W/2$. The final RGB head applies RMS normalization,

SiLU, a 3×3 convolution from 256 to 12 channels, and pixel shuffle. The stem, RGB head, and latent input/output projections do not use the stage-level shortcuts described above.

For f32, encoder shortcut group sizes are $g = (2, 1, 2, 4)$ from fine to coarse, and decoder repetition factors are $r = (4, 2, 4, 2)$ from coarse to fine. The additional f64 stage uses $g = r = 4$. All configurations use deterministic latents.

Distinction from pyramid outputs. GSC adds feature tensors within a backbone stage. In contrast, the pyramid output heads in Sec. 3.3 map intermediate decoder features to RGB predictions for PADT supervision. Both variants supervise the full-channel path with the original image. DC-AE 2.0-Structured also supervises pyramid outputs, whereas standard DC-AE 2.0 uses only the full-channel reconstruction loss.

For both f32 and f64, the four feature maps feeding the output heads have spatial sizes $H/16 \times W/16$, $H/8 \times W/8$, $H/4 \times W/4$, and $H/2 \times W/2$. Each RGB head follows the final head's RMS normalization–SiLU–3×3 convolution–pixel shuffle structure, mapping its input channels to 12 channels before $2\times$ upsampling. In both configurations, the predictions match pyramid targets at $H/8 \times W/8$, $H/4 \times W/4$, $H/2 \times W/2$, and $H \times W$. Thus, f64 adds an upsampling stage before the first output head, rather than adding a coarser supervision target. Each pyramid level uses a separable 5×5 binomial filter with 1D weights $[1, 4, 6, 4, 1]/16$, reflection padding, and stride-2 subsampling.

A.3 Three-Stage Training Recipe

Following the staged training recipes of DC-AE 1.0 (Chen et al., 2024b) and DC-AE 1.5 (Chen et al., 2025), we use a three-stage curriculum for all autoencoder configurations.

Both variants use the per-stage settings in Tab. 7, but DC-AE 2.0-Structured first inherits a completed standard Stage 1 checkpoint before its own three stages. All stages use AdamW with zero weight decay and an exponential moving average with target decay 0.9999 and exponential warmup with a 2,000-step time constant. The reconstruction objective assigns unit weight to both L1 and LPIPS, using a frozen VGG-based LPIPS network.

Table 7: Shared per-stage training settings for DC-AE 2.0 and DC-AE 2.0-Structured across all configurations. Autoencoder learning rates apply only to trainable parameters.

Setting	Stage 1	Stage 2	Stage 3
Image resolution	256×256	1024×1024	512×512
Training steps	500,000	60,000	500,000
Global batch size	512	128	512
Autoencoder learning rate	4.8×10^{-5}	1.6×10^{-5}	3.2×10^{-5}
Discriminator learning rate	–	–	2×10^{-3}
AdamW (β_1, β_2)	$(0.9, 0.95)$	$(0.9, 0.95)$	$(0, 0.99)$
Warmup steps	50,000	50,000	0
Initial warmup learning rate	4.8×10^{-6}	4.8×10^{-6}	–
L1 / LPIPS weights	$1/1$	$1/1$	$1/1$
Adversarial weight	0	0	0.5
Precision	BF16	BF16	FP32

Stage 1: reconstruction training. Standard DC-AE 2.0 jointly optimizes all encoder and decoder parameters at 256×256 for 500K steps using L1 and LPIPS losses. The learning rate increases linearly from 4.8×10^{-6} to 4.8×10^{-5} over 50,000 steps and remains constant thereafter.

Stage 2: high-resolution training. We train at 1024×1024 for 60K steps without adversarial loss. In both encoder and decoder, the residual stages at $H/8 \times W/8$ and coarser scales, together with their latent-side projections, remain trainable; the stem, final full-resolution RGB head, and higher-resolution stages are frozen. For DC-AE 2.0-Structured, all three auxiliary RGB heads remain trainable. The learning rate is warmed up linearly to 1.6×10^{-5} and then held constant.

Stage 3: adversarial decoder fine-tuning. We freeze the entire encoder and fine-tune the decoder at 512×512 for 500K steps. The decoder objective combines L1, LPIPS, and an adversarial loss

weighted by 0.5, using a StyleGAN2 discriminator. Decoder and discriminator updates alternate on successive training iterations, with discriminator regularization every 32 iterations. Their learning rates remain constant throughout this stage.

DC-AE 2.0-Structured. The Structured variant starts from the standard DC-AE 2.0 *Stage 1 checkpoint* of the same configuration. Structured Stage 1 then applies PADT at 256×256 for an additional 500K steps, supervising channel prefixes through hierarchical decoder outputs and matched Gaussian-pyramid targets. Structured Stage 2 continues PADT at 1024×1024 for 60K steps. Neither stage uses adversarial loss. Structured Stage 3 freezes the resulting encoder and follows the same 500K-step adversarial decoder fine-tuning procedure at 512×512 as standard DC-AE 2.0.

Each PADT batch uses the full channel budget c and three budgets sampled uniformly with replacement from $\mathcal{C}$. The loss averages over all four entries, including repeats, so $\mathcal{B}$ denotes a multiset.

A.4 Channel-Wise Frequency Analysis

For f32c128, we compare standard and Structured DC-AE 2.0 using latents extracted from the ImageNet test set. We analyze the spatial frequency content of individual latent channels with a blockwise discrete cosine transform (DCT). Each channel is standardized using its mean and standard deviation across the analyzed samples and spatial positions. We then apply a two-dimensional DCT to non-overlapping 8×8 blocks, discarding incomplete boundary blocks. For channel j, the plotted curve is

$$a_j(k) = \frac{1}{|\mathcal{W}|} \sum_{b \in \mathcal{W}} \frac{|D_{j,b}[\pi(k)]|}{\sum_{\ell=0}^{63} |D_{j,b}[\pi(\ell)]| + \epsilon}, \qquad k = 0, \ldots, 63, \tag{4}$$

where $D_{j,b}$ is the DCT of a standardized block, $\mathcal{W}$ contains blocks across samples, and $\epsilon = 10^{-8}$. The zigzag order π traverses increasing sums of horizontal and vertical frequency indices, starting with the DC term at $k = 0$. Normalization is performed before averaging, and the DC term is retained. Thus, the curves describe relative coefficient magnitudes, not absolute spectral energy.

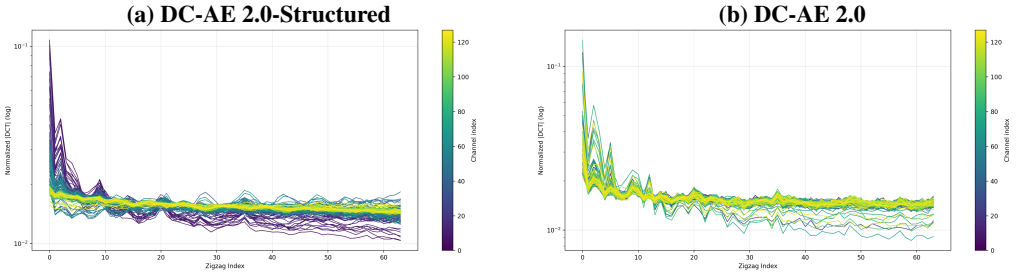

Figure 8: **Channel-wise latent frequency profiles on the ImageNet test set at f32c128.** Each curve follows Eq. 4; colors indicate the original channel order, from early (purple) to late (yellow). Both panels use logarithmic vertical axes with independently scaled limits.

In DC-AE 2.0-Structured, early channels devote more relative magnitude to low frequencies, while later channels have larger high-frequency shares. Standard DC-AE 2.0 has overlapping spectra without clear channel ordering. This supports the coarse-to-fine organization encouraged by PADT without implying disjoint frequency bands.

B Generation Post-Training

B.1 Overview and Training Recipe

We detail the generation post-training pipeline outlined in Sec. 3.4. It comprises diffusion adaptation followed by RL-based detail enhancement. The latter uses AnyFlow (Gu et al., 2026) flow-map training in Stage 1 to prepare the model for joint DMD and image-level reward optimization in Stage 2. Tab. 8 summarizes the training settings. All stages share the same training corpus, comprising 100K images in each of four categories: general-domain images, English text rendering, Chinese text rendering, and small faces (400K total). Section B.5 evaluates RL-based detail enhancement at a fixed 20-step inference budget.

Table 8: Shared generation post-training settings across the evaluated model and autoencoder configurations. All stages use AdamW with zero weight decay and BF16 mixed precision. Learning rates remain constant after warmup, when used.

	Diffusion adaptation				RL-based detail enhancement	
Setting	Phase 1	Phase 2	Phase 3	Phase 4	Stage 1	Stage 2
Image resolution	512^2	512^2	512^2	1024^2	1024^2	1024^2
Training steps	3,000	5,000	5,000	2,000	2,000	2,000
Global batch size	128	256	256	256	256	128
Learning rate	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-6}	10^{-6}
AdamW (β_1, β_2)	$(0.9, 0.95)$	$(0.9, 0.999)$	$(0.9, 0.95)$	$(0.9, 0.95)$	$(0.9, 0.95)$	$(0, 0.999)$
Warmup steps	0	1,000	1,000	1,000	1,000	0
Initial warmup LR	–	6.4×10^{-6}	6.4×10^{-6}	6.4×10^{-6}	0	–
EMA decay	–	–	0.999	0.999	0.999	0.99
LoRA rank / alpha	–	–	256/256	256/256	256/256	256/256
Gradient clipping	–	0.1	0.1	0.1	1.0	1.0

B.2 Diffusion Adaptation

Following DC-Gen (He et al., 2025), Phase 1 aligns the new patch embedder with frozen base-model embeddings. Phase 2 trains the embedder and output head while keeping the backbone frozen. Phase 3 additionally trains backbone LoRA adapters, and Phase 4 continues this adaptation at higher resolution. The autoencoder remains frozen throughout. For DC-AE 2.0-Structured, Phases 2–4 use channel-masked training (Chen et al., 2025), randomly selecting full channels or a partial channel prefix.

B.3 Any-Step Flow-Map Training

Stage 1 learns a flow map between arbitrary time pairs, enabling variable-step inference and preparing the model for reward optimization in Stage 2.

Let $(\mathbf{z}_0, \mathbf{y})$ denote a clean training latent and its conditioning input. We use the flow-matching parameterization (Lipman et al., 2022) with data at $t = 0$ and noise at $t = 1$. For $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $t \in [0, 1]$, the interpolated latent and target velocity are

$$\mathbf{z}_t = (1 - t)\mathbf{z}_0 + t\boldsymbol{\epsilon}, \qquad \mathbf{v}_t = \boldsymbol{\epsilon} - \mathbf{z}_0. \tag{5}$$

Following MeanFlow (Geng et al., 2025a), we parameterize an average velocity $\mathbf{u}_\theta$ over $[r, t]$ and define the transition

$$\mathbf{f}_\theta(\mathbf{z}_t, t, r) = \mathbf{z}_t - (t - r)\mathbf{u}_\theta(\mathbf{z}_t, t, r) \approx \mathbf{z}_r. \tag{6}$$

The boundary case $r = t$ recovers the instantaneous flow field, while arbitrary $r < t$ supports variable-step inference. Using the target velocity $\mathbf{v}_t$ defined in Eq. 5, we train with

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{t,r}\left[w_{t,r}\left\|\mathbf{u}_\theta(\mathbf{z}_t, t, r) - \text{sg}(\mathbf{u}_{\text{tgt}})\right\|_2^2\right], \quad \mathbf{u}_{\text{tgt}} = \mathbf{v}_t - (t - r)\frac{d}{dt}\mathbf{u}_\theta(\mathbf{z}_t, t, r). \tag{7}$$

Following AnyFlow (Gu et al., 2026), we approximate the total derivative with a symmetric finite difference, avoiding Jacobian–vector products. Half the batch uses $r = t$, a quarter uses $r = 0$, and the rest use arbitrary intervals. We normalize non-boundary losses against the $r = t$ examples and interpolate source- and target-time embeddings for transformer conditioning.

Implementation. Stage 1 initializes from the merged Phase 4 checkpoint and trains LoRA adapters and the added time embeddings. The EMA copies the current weights for the first 1,000 steps, then uses the decay in Tab. 8.

B.4 RL-Based Detail Enhancement with Joint Distillation

Stage 2 focuses on RL-based detail enhancement through image-level reward optimization, jointly with DMD and flow-map co-training.

On-policy flow-map distillation. Forward training does not directly correct errors accumulated on the model's own trajectories. We therefore use Distribution Matching Distillation (DMD) (Yin et al., 2024b;a) with AnyFlow's flow-map backward simulation (Gu et al., 2026). For a sampled S-step schedule $1 = t_0 > \cdots > t_S = 0$, we choose one interval and construct the endpoint with three differentiable flow-map transitions:

$$\hat{\mathbf{z}}_{t_k} = \mathbf{f}_\theta(\mathbf{z}_1, 1, t_k), \tag{8}$$

$$\hat{\mathbf{z}}_{t_{k+1}} = \mathbf{f}_\theta(\hat{\mathbf{z}}_{t_k}, t_k, t_{k+1}), \tag{9}$$

$$\hat{\mathbf{z}}_0 = \mathbf{f}_\theta(\hat{\mathbf{z}}_{t_{k+1}}, t_{k+1}, 0). \tag{10}$$

Varying S and k covers different inference budgets while keeping the simulation cost fixed.

We re-noise the self-rollout endpoint as

$$\tilde{\mathbf{z}}_s = (1 - s)\hat{\mathbf{z}}_0 + s\boldsymbol{\epsilon}, \tag{11}$$

and compare a frozen reference score $s_{\bar{\theta}}$ with a trainable fake score s_ϕ that tracks the student distribution. Both are initialized from Stage 1. The reverse-KL gradient is

$$\nabla_\theta \mathcal{L}_{\mathrm{DMD}} = -\mathbb{E}\left[\omega(s)\left(s_{\bar{\theta}}(\tilde{\mathbf{z}}_s, s, \mathbf{y}) - s_\phi(\tilde{\mathbf{z}}_s, s, \mathbf{y})\right)\frac{\partial \hat{\mathbf{z}}_0}{\partial \theta}\right], \tag{12}$$

Here $\omega(s)$ normalizes gradients; retaining $\mathcal{L}_{\mathrm{flow}}$ preserves the learned transition family.

Reward-tilted initialization and RL objective. Let $R(\hat{\mathbf{x}}, \mathbf{y})$ be a differentiable reward for $\hat{\mathbf{x}} = \mathcal{D}(\hat{\mathbf{z}}_0)$. Inspired by DMDR (Jiang et al., 2026), during a cold start, detached rewards only reweight per-sample DMD updates, favoring higher-reward rollouts without backpropagating through the reward model. We then remove this tilt and optimize

$$\mathcal{L}_{\mathrm{post}} = \mathcal{L}_{\mathrm{flow}} + \lambda_{\mathrm{DMD}}\mathcal{L}_{\mathrm{DMD}} - \lambda_{\mathrm{RL}}\mathbb{E}\left[R(\mathcal{D}(\hat{\mathbf{z}}_0), \mathbf{y})\right]. \tag{13}$$

The reward term directly optimizes generation behavior, while DMD constrains rollout-level distribution drift and flow-map co-training preserves any-step capability.

Implementation. Stage 2 initializes from the Stage 1 EMA checkpoint and updates the generator and fake-score LoRA adapters once per training step, while freezing the time embeddings and the reference model. The EMA copies the current weights for the first 200 steps, then uses the decay in Tab. 8. Stage 2 uses HPSv3++ (Liu et al., 2026) with a fixed iteration condition of zero and loss weights $\lambda_{\mathrm{DMD}} = \lambda_{\mathrm{RL}} = 0.5$.

The first 500 steps use reward-tilted DMD with temperature 1; subsequent steps enable direct reward gradients. Flow-map co-training remains active throughout, and rollout schedules are sampled from $\{2, 4, 8, 20\}$ steps. The data-loader batch size is 128; DMD and reward optimization use a 32-sample subset (two per GPU across 16 GPUs) both before and after cold start. The fake-score update draws a separate data batch and uses the same subset size.

B.5 Effect of RL-Based Detail Enhancement

Evaluation Protocol. We evaluate RL-based detail enhancement after adapting FLUX.2-dev to DC-AE 2.0 f32c32 and DC-AE 2.0-Structured f32c128. For each configuration, enhancement starts from the exact adaptation checkpoint used for the baseline and consists of AnyFlow Stage 1 followed by joint DMD and RL in Stage 2, with 2,000 updates per stage (Sections B.3 and B.4). All four evaluations use 20 sampling steps, with matched resolution, guidance, evaluation prompts, and sampling protocol. This paired comparison measures the effect of the complete enhancement procedure at a fixed inference budget.

Results Across Latent Configurations. As shown in Tab. 9, RL-based detail enhancement improves English and Chinese text rendering and all three AGHI-QA metrics for both configurations at the same 20-step inference budget. The f32c128 Structured configuration shows larger absolute gains in text rendering and FaceScore mean and P10, with the most pronounced recovery in the lower tail of face quality. Although it starts below f32c32 in both FaceScore metrics, it finishes above it after enhancement. The face detection rate increases by the same amount for both configurations.

Table 9: **Effect of RL-based detail enhancement after FLUX.2-dev autoencoder adaptation.** Enhancement comprises AnyFlow Stage 1 followed by joint DMD and RL in Stage 2. Each enhanced model continues from its paired adaptation checkpoint. All evaluations use 20 sampling steps and matched inference settings. Δ denotes the enhanced value minus the adaptation-only value, computed before rounding. Metrics retain their original scales.

Training stage	**LongText-Bench**		**AGHI-QA**		
	EN	ZH	FaceScore Mean ↑	FaceScore P10 ↑	Detection Rate ↑
DC-AE 2.0, f32c32					
Adaptation only	0.8990	0.8869	4.2789	1.9693	0.9575
+ RL-based detail enhancement	0.9666	0.9290	4.4589	2.1493	0.9700
Δ (after − before)	+0.0676	+0.0421	+0.1800	+0.1800	+0.0125
DC-AE 2.0-Structured, f32c128					
Adaptation only	0.7595	0.8008	4.0510	0.1581	0.9675
+ RL-based detail enhancement	0.9692	0.9467	4.5454	2.6187	0.9800
Δ (after − before)	+0.2097	+0.1459	+0.4944	+2.4606	+0.0125

C Additional Visual Comparisons

We compare original FLUX.2-dev with models adapted to DC-AE autoencoders, focusing on text rendering and facial details. Each row uses a matched prompt, 1024×1024 resolution, and 20 sampling steps. Red boxes mark regions enlarged at the same scale within each row. In these examples, DC-AE 2.0-Structured f32c128 generally produces finer text and facial details than DC-AE 2.0 f32c32, which in turn improves upon DC-AE 1.0 f32c32.



Figure 9: **Text rendering on LongText-Bench (English).**



Figure 10: **Text rendering on LongText-Bench (Chinese).**



Figure 11: **Facial details on AGHI-QA.**