Efficient Fréchet Distributional Decoder Alignment for Latent Generative Models

Xianghong Fang[1,*] **Jiajun Zhu**[2,*] **Esma Aimeur**[2]
Dehan Kong[1] **Tim G. J. Rudner**[1,3]

[1]University of Toronto [2]Université de Montréal [3]Vijil

Website Code & Models

Abstract

Fréchet Distributional (FD) objectives provide an effective distribution-level signal for post-training generative models. For latent generative models, however, existing FD-based post-training methods optimize only the latent generator while keeping the decoder frozen: the decoder is trained on reconstruction latents obtained from real images but deployed on generator-produced latents at generation time. When these latent distributions differ, this **decoder train-generation mismatch** means that the decoder is deployed on inputs drawn from a distribution it was not directly trained on. We introduce **Fréchet Distributional Decoder Alignment (FDDA)**, an efficient post-training method that freezes the latent generator and directly optimizes the decoder on generation-time latents using an FD objective. A single epoch of FDDA consistently improves gFID and gFDr^6 across diverse flow-based and autoregressive latent generative models. Across all 11 evaluated models, FDDA reduces gFDr^6 by 34.0-59.6% from the original pretrained models and provides a further 10.2-57.2% reduction after generator-side FD post-training. FDDA also reduces total wall-clock time by factors of $3.6\times$-$328.7\times$ relative to generator-side FD post-training across these models. These results show that generator-side post-training alone does not exhaust the benefits of FD optimization, and that further gains can be obtained by adapting the decoder at substantially lower cost.

1 Introduction

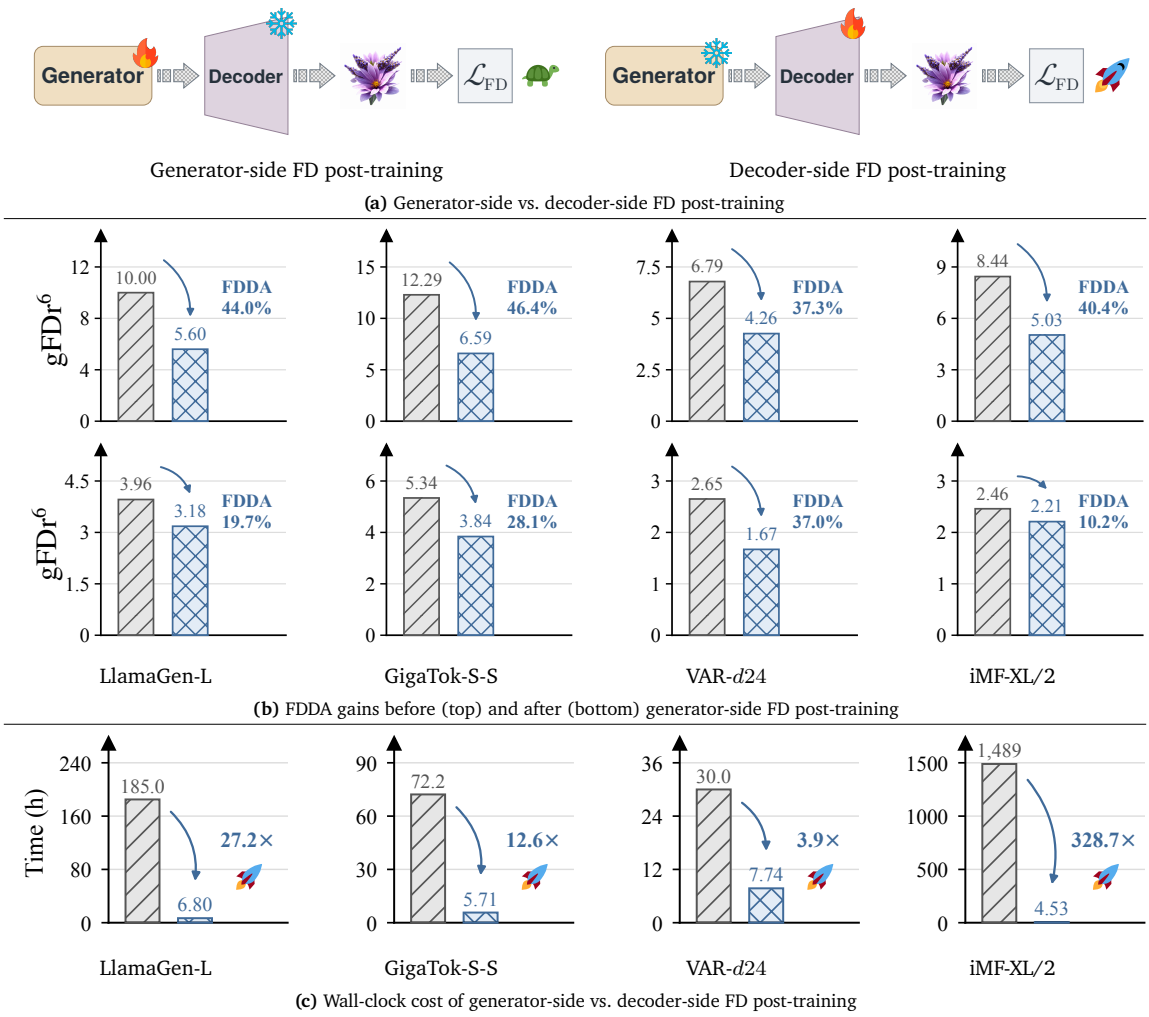
Post-training has become an effective way to further improve pretrained generative models without retraining them from scratch [Xu+23; Bla+24; Luo+25; Fan+26; Lin+26]. In particular, Fréchet Distributional (FD) objectives match the representation-space statistics of generated and real samples, providing a distribution-level objective for post-training [Yan+26a]. Recent FD-based post-training methods have demonstrated substantial improvements in generation quality across diverse generative model families [Yan+26a; Zha+26; Gao+26]. Yet in latent generative models, these methods optimize only the generator; the decoder has not been studied as a target for distribution-level post-training [Yan+26a; Zha+26].

This raises a natural question:

Does improving the generator exhaust the potential of distribution-level post-training?

For latent generative models, two observations suggest otherwise. First, generation relies on two components: Alatent generator that produces a generation-time latent $z_g \sim p_g(z)$, and a decoder that maps it to an image [OVK17; ERO21; Rom+22; Sun+24]. Since both components affect final image quality, the decoder can also be optimized with a distribution-level objective. Second, the decoder

[*] Equal contribution.



(a) Generator-side vs. decoder-side FD post-training

(b) FDDA gains before (top) and after (bottom) generator-side FD post-training

(c) Wall-clock cost of generator-side vs. decoder-side FD post-training

Figure 1: Decoder-side FD post-training with FDDA. (a) Existing methods optimize the latent generator with a frozen decoder, whereas FDDA freezes the generator and optimizes the decoder on generation-time latents. **(b)** FDDA provides complementary gains. Top row: FDDA applied to released checkpoints before generator-side FD post-training. Bottom row: FDDA applied after generator-side FD post-training. Gray and blue bars denote gFDr^6 before and after FDDA, and annotations report the relative reduction. **(c)** Decoder-side FD post-training substantially reduces training cost. Gray (left) and blue (right) bars denote total wall-clock hours on a single H100 GPU for generator-side and decoder-side FD post-training, and annotations show the reduction factor.

faces a **train-generation mismatch**: It is typically trained on reconstruction latents $z_r \sim p_r(z)$ obtained by encoding real images [OVK17; ERO21; Rom+22], yet deployed on generation-time latents z_g produced by the generator. When $p_r(z)$ and $p_g(z)$ differ, the decoder receives generation-time inputs that it was not directly trained on [Qiu+25; Fan+26]. Generator-side post-training does not directly adapt the decoder to its generation-time inputs: it updates the generator and hence $p_g(z)$, while the decoder remains optimized primarily on $p_r(z)$.

To address this mismatch, we introduce **Fréchet Distributional Decoder Alignment (FDDA)**, which extends FD-based post-training to the decoder, as illustrated in Figure 1a. FDDA freezes the pretrained latent generator and optimizes the decoder on generation-time latents $z_g \sim p_g(z)$ using an FD objective on the generated images. In this way, FDDA adapts the decoder to the latent distribution it receives during generation while leaving the pretrained generator unchanged, providing a decoder-side complement to generator-side FD post-training.

FDDA improves generation quality across both flow-based and autoregressive latent generative models.

Across all 11 evaluated configurations, FDDA improves both gFID [Heu+17] and gFDr^6 [Yan+26a] when applied to the original pretrained models. For the four configurations shown in Figure 1b, gFDr^6 decreases by 44.0% on LlamaGen-L [Sun+24], 46.4% on GigaTok-S-S [Xio+25], 37.3% on VAR-$d24$ [Tia+24], and 40.4% on iMF-XL/2 [Gen+26]. The gains remain after generator-side FD post-training: FDDA further reduces gFDr^6 by 19.7%, 28.1%, 37.0%, and 10.2% on these four configurations, respectively, while improving gFDr^6 across all 11 evaluated configurations. FDDA also reduces the wall-clock cost of FD post-training. As shown in Figure 1c, its total wall-clock time is lower by factors of $27.2\times$, $12.6\times$, $3.9\times$, and $328.7\times$ for the LlamaGen, GigaTok, VAR, iMF configurations, respectively. These results show that generator-side post-training alone does not exhaust the benefits of FD optimization, as further gains can still be obtained by adapting the decoder at substantially lower cost.

Our contributions are threefold:

1. We introduce **Fréchet Distributional Decoder Alignment (FDDA)**, which freezes the latent generator and optimizes the decoder on generation-time latents with an FD objective, directly addressing the decoder train-generation mismatch.
2. We show that a single epoch of FDDA improves gFID and gFDr^6 across all 11 evaluated flow-based and autoregressive latent generative models, while requiring substantially less total wall-clock time than generator-side FD post-training.
3. We show that FDDA provides further gFDr^6 improvements after generator-side FD post-training across all evaluated models, indicating that generator-side optimization alone does not exhaust the benefits of FD post-training.

2 Related Work

Post-training generative models. Post-training has emerged as an effective approach for improving pretrained generative models with objectives that differ from those used during pretraining. Prior work has optimized diffusion models toward downstream rewards or human preferences. Reward-based approaches include direct reward feedback [Xu+23], reinforcement-learning-based fine-tuning [Bla+24; Fan+23], direct back-propagation through differentiable rewards [Cla+24], and scalable reward optimization [Den+24]. Preference-based methods instead directly fine-tune pretrained diffusion models from pairwise feedback [Wal+24; Yan+24]. Other work has explored adversarial objectives [Lin+26] and distribution-level objectives [Yan+26a; Zha+26; Gao+26] for post-training pretrained generators. Our work follows this general direction, but applies distribution-level post-training to the decoder rather than further optimizing the latent generator.

Fréchet distributional post-training. Fréchet Inception Distance (FID) was introduced as an evaluation metric for generative models [Heu+17]. Earlier work explored feature-moment matching and direct optimization of Fréchet-style objectives [MH21; Doa+20], but these methods were limited by batch-scale statistics. FD loss [Yan+26a] makes large-population FD optimization practical by decoupling the statistics used for FD estimation from the batch used for gradient computation, making it suitable for post-training pretrained visual generators. Subsequent work extends FD-based post-training to autoregressive image generators [Zha+26] and introduces adversarially learned representation spaces for FD optimization [Gao+26]. Existing scalable FD-based post-training methods, however, optimize the generator while keeping the decoder fixed in latent generative models. FDDA instead applies FD-based post-training directly to the decoder on generation-time latents.

Decoder alignment with generation-time latents. Latent generative models typically train their tokenizer or autoencoder separately from the downstream generator [OVK17; ERO21; Rom+22]. The decoder is therefore trained on reconstruction latents but used on generator-produced latents at generation time, resulting in a decoder train-generation mismatch [Qiu+25; Fan+26]. Prior work has reduced this mismatch by adapting the decoder using latents that more closely resemble

those encountered during generation. RobusTok [Qiu+25] performs generator-aware decoder post-training and, for autoregressive generators, introduces a preservation ratio that retains part of the reconstruction tokens during sampling. Generation-aware reconstruction [Fan+26] constructs a reconstruction-to-generation trajectory and adapts the decoder on intermediate generative latents while preserving correspondence with source images. These methods use paired supervision and retain per-sample reconstruction targets during decoder adaptation. FDDA instead optimizes the decoder directly on fully generated latents $z_g \sim p_g(z)$ with an FD objective while keeping the latent generator fixed. Its distribution-level objective does not require per-sample correspondence between generated and real images.

3 Preliminaries and Problem Setup

3.1 Latent Generative Models

Latent generative models typically follow a two-stage training paradigm [OVK17; ERO21; Rom+22; Sun+24]. First, a tokenizer or autoencoder learns a latent representation of images. Given $x \sim p_{\text{data}}(x)$, the encoder produces a reconstruction latent $z_r = \mathcal{E}_\theta(x) \sim p_r(z)$, from which the decoder $\mathcal{D}_\phi$ is trained to reconstruct x. Here, $p_r(z)$ denotes the encoder-induced distribution over decoder inputs. A latent generator is then trained over this latent space and, at generation time, produces latents $z_g \sim p_g(z)$ that are decoded as $x_g = \mathcal{D}_\phi(z_g)$.

We use z_g to denote the generation-time input to the decoder regardless of the generator type. For diffusion- and flow-based latent generators, z_g is obtained from the final state of the sampling process [Rom+22; PX23; Gen+25; Gen+26]. For autoregressive latent generators, the generator first samples discrete tokens, which are then converted through the tokenizer codebook into generation-time latents z_g [ERO21; Sun+24; Xio+25; Yu+24; Tia+24]. We use $p_g(z)$ to denote the resulting distribution over decoder inputs in both settings.

The decoder is therefore trained on reconstruction latents $z_r \sim p_r(z)$ but used on generation-time latents $z_g \sim p_g(z)$. When $p_r(z)$ and $p_g(z)$ differ, the decoder is applied to inputs drawn from a distribution on which it was not directly trained. We refer to this as the **decoder train-generation mismatch** [Qiu+25; Fan+26].

3.2 Fréchet Distributional Objectives

Fréchet Distance (FD) measures distributional discrepancy by comparing the first- and second-order statistics of real and generated samples in a representation space [Heu+17; Yan+26a]. Let $\psi(\cdot)$ denote a representation model. For real images $x \sim p_{\text{data}}(x)$ and generated images x_g, let μ_r and Σ_r denote the mean and covariance of $\psi(x)$, and let μ_g and Σ_g denote the corresponding statistics of $\psi(x_g)$. The Fréchet distance is

$$\text{FD}_\psi = \|\mu_r - \mu_g\|_2^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r^{1/2}\Sigma_g\Sigma_r^{1/2}\right)^{1/2}\right). \tag{3.1}$$

The choice of ψ specifies the feature space of the FD objective: Inception-v3 [Sze+16] recovers the standard Fréchet Inception Distance (FID) [Heu+17], while other pretrained representations define FD objectives in their respective feature spaces [Yan+26a].

Directly optimizing FD is challenging because estimating its population statistics requires substantially more samples than are available in a typical minibatch. FD-loss [Yan+26a] addresses this issue by decoupling the population used for FD estimation from the minibatch used for gradient computation, making large-population FD optimization practical for post-training diffusion- and flow-based generative models. Zhang et al. [Zha+26] extend FD post-training to discrete autoregressive generators via rollout-based optimization and straight-through estimation. AdvFD [Gao+26] further learns the feature space adversarially to expose remaining discrepancies between real and generated distributions.

Existing FD-based post-training methods are *generator-side*, optimizing the generator while keeping the decoder fixed. In latent generative models, they therefore do not directly adapt the decoder to its generation-time inputs. We instead apply FD optimization to the decoder on generation-time latents, as described in the next section.

4 Fréchet Distributional Decoder Alignment

4.1 Decoder-Side FD Post-Training

We introduce **Fréchet Distributional Decoder Alignment (FDDA)**, which extends FD post-training from the generator to the decoder. Given a pretrained latent generator with generation-time latent distribution $p_g(z)$, FDDA freezes the generator and updates only the decoder $\mathcal{D}_\phi$. Specifically, generation-time latents $z_g \sim p_g(z)$ are sampled from the frozen generator and decoded as $x_g = \mathcal{D}_\phi(z_g)$. This defines the generated image distribution $p_\phi^x = (\mathcal{D}_\phi)_\# p_g$, where p_ϕ^x is the distribution of generated images obtained by decoding $z_g \sim p_g(z)$. FDDA optimizes the decoder by matching p_ϕ^x to the real data distribution:

$$\phi^\star = \arg\min_\phi \mathrm{FD}_\psi \left(p_{\mathrm{data}}, p_\phi^x\right), \qquad p_g \text{ fixed.} \tag{4.1}$$

Generator-side FD post-training updates the generator and hence changes $p_g(z)$. FDDA instead keeps $p_g(z)$ fixed and trains the decoder directly on samples from the latent distribution encountered at generation time, reducing the mismatch between decoder training and generation.

FDDA uses distribution-level supervision and requires neither reconstruction targets nor per-sample correspondence between generated and real images. This differs from paired decoder adaptation methods [Qiu+25; Fan+26], which retain sample-level reconstruction supervision. FDDA directly trains the decoder on fully generated latents $z_g \sim p_g(z)$. The formulation is the same for diffusion- and flow-based and autoregressive latent generators; only the procedure used to obtain z_g changes. Gradients in FDDA pass only through the decoder. The latent generator and its sampling process are not part of the backward pass, which avoids the cost of differentiating through multi-step sampling or autoregressive rollouts during generator-side FD post-training.

4.2 Multi-Representation FD Optimization

FD can be computed in different representation spaces, each capturing different aspects of distributional discrepancy. Following FD-loss [Yan+26a], we adopt six fixed pretrained feature extractors: Inception-v3 [Sze+16], ConvNeXt-v2 [Woo+23], DINOv2 [Oqu+24], MAE [He+22], SigLIP2 [Tsc+25], and CLIP [Rad+21]. The FDDA training objective uses the subset $\Psi_{\text{train}} = \{\text{Inception-v3}, \text{MAE}, \text{SigLIP2}\}$, while the remaining three representations are held out for evaluation in all subsequent experiments.

FD values can differ substantially in scale across representation spaces, so directly summing them can cause terms with larger numerical values to dominate the objective. We normalize each representation-specific FD by its stop-gradient value. At iteration t, FDDA optimizes

$$\mathcal{L}_{\text{FDDA}} = \sum_{\psi_k \in \Psi_{\text{train}}} w_k \frac{\mathrm{FD}_{\psi_k}^{(t)}}{\mathrm{sg}\left(\mathrm{FD}_{\psi_k}^{(t)}\right) + c}, \tag{4.2}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient, $c = 0.01$ is a small constant for numerical stability, and $w_k = 1$ for all representations. The denominator is stop gradient, and serves only as a scale factor during back-propagation, so normalization does not change the gradient direction of each FD term. This reduces the effect of differences in numerical scale across representation spaces.

Estimating the generated feature statistics in Equation 4.2 requires more samples than are available in a single minibatch. We use an EMA-based estimator for this purpose. For each representation ψ_k,

the real-image statistics $(\mu_{r,k}, \Sigma_{r,k})$ are precomputed once and kept fixed. The EMA statistics $\mu_{g,k}^{(0)}$ and $M_{g,k}^{(0)}$ are initialized from $50{,}000$ samples drawn from the frozen generator prior to training. At iteration t, given a minibatch of generation-time latents $\{z_{g,i}^{(t)}\}_{i=1}^{B}$ sampled from the frozen generator, we compute the generated features $h_{k,i}^{(t)} = \psi_k(\mathcal{D}_\phi(z_{g,i}^{(t)}))$. Let $\mu_{b,k}^{(t)} = \frac{1}{B}\sum_{i=1}^{B} h_{k,i}^{(t)}$ and $M_{b,k}^{(t)} = \frac{1}{B}\sum_{i=1}^{B} h_{k,i}^{(t)} h_{k,i}^{(t)\top}$ denote the batch mean and second moment, respectively. The running generated statistics are updated by

$$\begin{aligned}
\mu_{g,k}^{(t)} &= \beta \mu_{g,k}^{(t-1)} + (1-\beta)\mu_{b,k}^{(t)}, \\
M_{g,k}^{(t)} &= \beta M_{g,k}^{(t-1)} + (1-\beta)M_{b,k}^{(t)}, \\
\Sigma_{g,k}^{(t)} &= M_{g,k}^{(t)} - \mu_{g,k}^{(t)} \mu_{g,k}^{(t)\top},
\end{aligned}$$

where $\beta = 0.999$ is the EMA decay rate. The representation-specific $\mathrm{FD}_{\psi_k}^{(t)}$ in Equation 4.2 is then computed using the fixed real statistics $(\mu_{r,k}, \Sigma_{r,k})$ and the current generated statistics $(\mu_{g,k}^{(t)}, \Sigma_{g,k}^{(t)})$. EMA statistics carried over from previous iterations are treated as constants during back-propagation, so gradients flow only through features computed from the current minibatch. The latent generator and all representation models remain frozen throughout post-training, and only the decoder parameters ϕ are optimized. The EMA estimator avoids storing a large feature population while maintaining running statistics for the FD objective.

5 Experiments

We conduct extensive experiments to evaluate the **effectiveness**, **complementarity**, and **computational efficiency** of FDDA across diverse autoregressive and flow-based latent generative models. We evaluate FDDA on pretrained models and examine whether it provides further gains after generator-side FD post-training. We next compare its end-to-end post-training cost with generator-side FD post-training. Additional analyses cover longer training, reconstruction performance, and qualitative results. Code and checkpoints will be publicly released to ensure reproducibility.

5.1 Experimental Setup

Models and Training Setup. We evaluate FDDA on a diverse set of pretrained latent generative models. For autoregressive generation, we consider LlamaGen-B and LlamaGen-L [Sun+24], GigaTok-S-S [Xio+25], TiTok-L-32 and TiTok-B-64 [Yu+24], and VAR-$d16$, VAR-$d20$, and VAR-$d24$ [Tia+24]. For flow-based generation, we consider iMF-B, iMF-L, and iMF-XL [Gen+26]. We apply FDDA to both the original pretrained checkpoints and their generator-side FD-post-trained counterparts from Zhang et al. [Zha+26] for autoregressive models and Yang et al. [Yan+26a] for flow-based models. FDDA is implemented in two stages: we first sample generation-time latents from the frozen generator and cache them as a training pool, and then optimize only the decoder on the cached latents using the FDDA objective. Unless otherwise specified, all FDDA models are trained for one epoch. Additional implementation and training details are provided in Appendix A.

Evaluation Metrics. We primarily evaluate generation performance using gFDr^6 [Yan+26a], Fréchet Inception Distance [gFID; Heu+17], representation-specific Fréchet distances computed in ConvNeXt-v2 [Woo+23], DINOv2 [Oqu+24], MAE [He+22], SigLIP2 [Tsc+25], and CLIP [Rad+21] feature spaces ($\mathrm{gFD}_{\mathrm{conv}}$, $\mathrm{gFD}_{\mathrm{dino}}$, $\mathrm{gFD}_{\mathrm{mae}}$, $\mathrm{gFD}_{\mathrm{sig}}$, and $\mathrm{gFD}_{\mathrm{clip}}$), and generation Inception Score [gIS; Sal+16]. ConvNeXt-v2, DINOv2, and CLIP are held out from FDDA training and used only for evaluation. We also evaluate reconstruction performance using peak signal-to-noise ratio (PSNR), structural similarity index (SSIM), Fréchet Inception Distance [rFID; Heu+17], and perceptual similarity [LPIPS; Zha+18].

Table 1: FDDA before generator-side FD post-training. We apply FDDA to the original pretrained latent generative models. All metrics are computed using the same evaluation protocol following Yang et al. [Yan+26a]. ✗and ✓indicate whether each component undergoes FD post-training.

Models	Generator FD	Decoder FD	gFID (↓)	gIS (↑)	gFD_{conv} (↓)	gFD_{dino} (↓)	gFD_{mae} (↓)	gFD_{sig} (↓)	gFD_{clip} (↓)	$gFDr^6$(↓)	
Discrete autoregressive generative models											
LlamaGen-B	✗	✗	5.17	196.1	233.91	244.12	0.528	21.03	149.44	16.48	
	✗	✓	2.74	209.3	237.39	190.90	0.250	11.50	115.52	10.88	↓34.0%
LlamaGen-L	✗	✗	4.23	290.1	113.61	121.41	0.383	12.20	97.13	10.00	
	✗	✓	1.87	303.5	103.27	87.98	0.109	5.19	73.07	5.60	↓44.0%
GigaTok-S-S	✗	✗	4.34	268.5	135.95	182.66	0.432	15.25	112.26	12.29	
	✗	✓	1.68	281.6	99.96	123.43	0.169	6.32	74.75	6.59	↓46.4%
TiTok-L-32	✗	✗	2.75	194.1	190.86	192.05	0.567	18.86	157.93	15.32	
	✗	✓	1.34	202.6	181.26	116.29	0.124	5.49	70.83	6.19	↓59.6%
TiTok-B-64	✗	✗	2.44	206.6	174.01	197.82	0.509	17.36	154.73	14.57	
	✗	✓	1.61	214.9	184.09	129.98	0.157	6.18	80.39	6.99	↓52.0%
VAR-*d*16	✗	✗	3.27	280.8	148.97	150.36	0.428	13.56	104.23	11.12	
	✗	✓	1.62	280.1	152.45	121.49	0.175	6.68	81.14	7.03	↓36.8%
VAR-*d*20	✗	✗	2.61	305.2	111.78	102.19	0.344	9.84	85.74	8.46	
	✗	✓	1.32	296.9	124.40	82.77	0.127	4.95	65.96	5.34	↓36.9%
VAR-*d*24	✗	✗	2.11	314.7	94.78	76.34	0.286	7.37	73.93	6.79	
	✗	✓	1.24	304.4	112.00	60.89	0.097	3.75	55.13	4.26	↓37.3%
Flow-match generative models, one-step											
iMF-B	✗	✗	3.49	253.7	188.08	206.17	0.487	19.65	156.14	15.40	
	✗	✓	1.65	268.1	115.47	144.39	0.232	8.10	105.06	8.54	↓44.5%
iMF-L	✗	✗	1.88	275.4	105.53	111.56	0.282	11.30	101.18	9.11	
	✗	✓	1.26	281.0	75.93	83.77	0.137	5.12	72.12	5.48	↓39.8%
iMF-XL	✗	✗	1.81	281.9	97.57	103.01	0.261	10.37	94.58	8.44	
	✗	✓	1.14	288.3	67.12	77.30	0.117	4.67	67.74	5.03	↓40.4%

Table 2: FDDA after generator-side FD post-training. We apply FDDA to latent generative models whose generators have already undergone FD post-training. All metrics are computed using the same evaluation protocol following Yang et al. [Yan+26a]. ✗and ✓indicate whether each component undergoes FD post-training.

Models	Generator FD	Decoder FD	gFID (↓)	gIS (↑)	gFD_{conv} (↓)	gFD_{dino} (↓)	gFD_{mae} (↓)	gFD_{sig} (↓)	gFD_{clip} (↓)	$gFDr^6$ (↓)	
Discrete autoregressive generative models											
LlamaGen-B	✓	✗	2.45	277.9	118.37	105.41	0.174	6.07	70.55	6.33	
	✓	✓	2.23	275.9	134.35	95.79	0.100	4.55	61.77	5.27	↓16.7%
LlamaGen-L	✓	✗	1.32	315.9	74.54	57.09	0.104	3.18	54.27	3.96	
	✓	✓	1.34	313.5	83.61	49.14	0.052	2.40	44.71	3.18	↓19.7%
GigaTok-S-S	✓	✗	2.52	291.6	97.96	82.97	0.219	4.45	57.20	5.34	
	✓	✓	1.74	296.5	82.50	72.77	0.088	2.99	46.12	3.84	↓28.1%
TiTok-L-32	✓	✗	2.25	211.7	133.58	143.61	0.334	12.24	121.85	10.69	
	✓	✓	1.35	215.1	161.52	103.08	0.105	4.65	64.74	5.49	↓48.6%
TiTok-B-64	✓	✗	1.45	238.7	102.44	132.47	0.300	9.87	111.45	9.29	
	✓	✓	1.42	240.3	148.23	99.99	0.113	4.50	65.35	5.42	↓41.7%
VAR-*d*16	✓	✗	1.85	307.9	80.75	47.20	0.154	3.30	38.19	3.65	
	✓	✓	1.35	307.3	77.91	45.42	0.040	1.92	34.22	2.62	↓28.2%
VAR-*d*20	✓	✗	1.64	309.4	62.35	31.78	0.596	2.27	31.10	4.63	
	✓	✓	1.08	308.1	76.47	28.58	0.026	1.28	28.03	1.98	↓57.2%
VAR-*d*24	✓	✗	1.51	320.7	55.29	26.84	0.227	1.33	25.09	2.65	
	✓	✓	1.09	308.3	59.35	24.81	0.022	0.96	24.54	1.67	↓37.0%
Flow-matching generative models, one-step											
iMF-B	✓	✗	0.86	307.1	65.97	89.05	0.154	5.46	70.35	5.57	
	✓	✓	0.91	304.0	74.66	80.48	0.100	3.91	59.69	4.54	↓18.5%
iMF-L	✓	✗	0.77	303.4	43.60	38.89	0.050	2.46	38.85	2.72	
	✓	✓	0.80	299.1	65.13	36.92	0.035	2.05	33.72	2.43	↓10.7%
iMF-XL	✓	✗	0.76	306.7	40.34	33.85	0.043	2.21	35.74	2.46	
	✓	✓	0.78	303.7	57.53	32.51	0.030	1.82	31.61	2.21	↓10.2%

5.2 Effectiveness and Complementarity of FDDA

We evaluate FDDA under two complementary settings. We first apply FDDA to the original pretrained models to isolate decoder-side FD post-training. We then apply FDDA after generator-side FD post-training to test whether further gains remain. Finally, we compare representative FDDA models with a broader set of generative systems.

Table 3: System-level comparison on class-conditional ImageNet 256×256. NFE denotes the number of function evaluations during inference, and #Params denotes the number of parameters in the latent generator. All results use CFG where applicable; $\times 2$ indicates the additional function evaluations for CFG. The best result for each metric is highlighted in bold. †: results reported in Tables 4 and F.7 of FD-Loss [Yan+26a]. ‡: results measured by us following the evaluation protocol of Yang et al. [Yan+26a]. Decoder-side post-training is applied to the generator-side FD-post-trained models (FDDA+). *: CFG is applied only over a time subinterval; NFE is reported as the full-CFG upper bound.

Models	NFE	#params	gFID (↓)	gIS (↑)	gFD_{conv} (↓)	gFD_{dino} (↓)	gFD_{mae} (↓)	gFD_{sig} (↓)	gFD_{clip} (↓)	gFDr^6 (↓)
Without FD post-training										
LlamaGen-L‡	256×2	343M	4.23	290.1	113.61	121.41	0.383	12.20	97.13	10.00
GigaTok-S-S‡	256×2	111M	4.34	268.5	135.95	182.66	0.432	15.25	112.26	12.29
TiTok-L-32‡	8×2	177M	2.75	194.1	190.86	192.05	0.567	18.86	157.93	15.32
VAR-$d24$‡	10×2	1.0B	2.11	314.7	94.78	76.34	0.286	7.37	73.93	6.79
VAR-$d30$†	10×2	2B	1.97	304.6	96.57	75.35	0.29	7.19	74.61	6.70
BAR-L†	$256\times2_{\times4}$	1.1B	1.01	281.9	44.55	46.68	0.18	3.99	33.52	3.57
SiT-XL/2†	250×2	675M	2.12	256.7	114.89	111.86	0.24	9.76	99.16	8.44
MAR-L†	$256\times2_{\times100}$	478M	1.80	293.4	62.33	86.32	0.19	7.66	82.81	6.68
FlowAR-H†	50×2*	1.9B	1.68	274.1	73.68	66.42	0.20	7.71	69.99	6.13
MAR-H†	$256\times2_{\times100}$	942M	1.56	299.5	54.25	70.25	0.16	6.09	72.99	5.61
MAR-L, DeTok†	$256\times2_{\times100}$	478M	1.39	306.2	77.56	66.14	0.19	5.79	67.94	5.49
REG†	250×2*	685M	1.54	302.9	64.86	48.93	0.13	5.09	60.87	4.64
SiT-XL/2-REPA†	250×2*	675M	1.42	306.1	69.36	60.62	0.17	5.97	70.90	5.45
LightningDiT†	250×2	675M	1.42	294.3	62.19	53.38	0.13	5.12	57.24	4.57
DDT-XL†	250×2	675M	1.26	309.3	58.25	60.39	0.18	6.15	77.70	5.70
REPA-E†	250×2*	676M	1.17	298.3	72.89	34.57	0.11	3.05	35.18	3.04
RAE-XL†	50×2*	839M	1.16	261.0	101.72	29.92	0.14	2.29	44.13	3.26
Drift-L†	1	463M	1.53	257.2	115.37	146.88	0.28	14.59	121.01	10.92
iMF-XL‡	1	610M	1.81	281.9	97.57	103.01	0.261	10.37	94.58	8.44
iMF-XL†	2	610M	1.61	289.1	87.79	89.51	0.24	9.02	86.98	7.48
Generator-side FD post-training										
LlamaGen-L‡	256×2	343M	1.32	315.9	74.54	57.09	0.104	3.18	54.27	3.96
GigaTok-S-S‡	256×2	111M	2.52	291.6	97.96	82.97	0.219	4.45	57.20	5.34
TiTok-L-32‡	8×2	177M	2.25	211.7	133.58	143.61	0.334	12.24	121.85	10.69
VAR-$d24$‡	10×2	1.0B	1.51	**320.7**	55.29	26.84	0.227	1.33	25.09	2.65
iMF-XL‡	1	610M	**0.76**	306.7	**40.34**	33.85	0.043	2.21	35.74	2.46
Decoder-side FD post-training (Ours)										
LlamaGen-L‡	256×2	343M	1.34	313.5	83.61	49.14	0.052	2.40	44.71	3.18
GigaTok-S-S‡	256×2	111M	1.74	296.5	82.50	72.77	0.088	2.99	46.12	3.84
TiTok-L-32‡	8×2	177M	1.35	215.1	161.52	103.08	0.105	4.65	64.74	5.49
VAR-$d24$‡	10×2	1.0B	1.09	308.3	59.35	**24.81**	**0.022**	**0.96**	**24.54**	**1.67**
iMF-XL‡	1	610M	0.78	303.7	57.53	32.51	0.030	1.82	31.61	2.21

Before Generator-side FD Post-training. Table 1 evaluates FDDA on the original pretrained models. Across all 11 autoregressive and flow-based models, FDDA consistently improves both gFID and gFDr^6, with gFDr^6 reductions ranging from 34.0-59.6%. Improvements are also observed across multiple representation spaces, including gFD_{dino}, gFD_{mae}, gFD_{sig}, and gFD_{clip}. For example, on TiTok-L-32, FDDA reduces gFID from 2.75 to 1.34 and gFDr^6 from 15.32 to 6.19; on iMF-XL, gFID decreases from 1.81 to 1.14 and gFDr^6 from 8.44 to 5.03. These results show that directly adapting the decoder to generation-time latents provides substantial gains without modifying the pretrained latent generator.

After Generator-side FD Post-training. We next examine whether FDDA remains effective after the generator has already undergone FD post-training. As shown in Table 2, FDDA further reduces gFDr^6 across all 11 models, with improvements ranging from 10.2% to 57.2%. Improvements are also observed for gFD_{dino}, gFD_{mae}, gFD_{sig}, and gFD_{clip}, despite the substantially stronger starting points provided by generator-side FD post-training. Notably, gFDr^6 decreases from 10.69 to 5.49 on TiTok-L-32 and from 4.63 to 1.98 on VAR-$d20$, corresponding to reductions of 48.6% and 57.2%, respectively. Although improvements in individual metrics such as gFID are not uniform in this stronger setting, the consistent gains across the aggregate and multiple representation-space metrics indicate that decoder-side FD post-training remains complementary to generator-side optimization.

Table 4: Computational efficiency of FDDA. We compare the wall-clock time of generator-side FD post-training and FDDA on a single H100 GPU. All FDDA models are trained for one epoch, with total time including both generation-time latent pool construction and decoder training. Cost reduction is computed as the ratio of generator-side FD training time to the total FDDA time.

Models	Generator-side FD			Decoder-side FD (FDDA)				Cost Reduction
	Epochs	# Generator Params	Train (h)	# Decoder Params	Latent Pool (h)	Train (h)	Total (h)	
Discrete autoregressive generative models								
LlamaGen-B	1.25	111M	100.6	42.5M	1.50	2.77	4.27	23.6 ×
LlamaGen-L	1.25	343M	185.0	42.5M	4.02	2.78	6.80	27.2 ×
GigaTok-S-S	1.25	111M	72.2	83.8M	2.68	3.03	5.71	12.6 ×
TiTok-L-32	1.25	177M	16.7	337.6M	0.29	3.19	3.48	4.8 ×
TiTok-B-64	1.25	177M	16.1	118.8M	0.60	2.75	3.35	4.8 ×
VAR-$d16$	1.25	310M	18.9	64.7M	1.62	3.59	5.21	3.6 ×
VAR-$d20$	1.25	600M	24.4	64.7M	2.85	3.59	6.44	3.8 ×
VAR-$d24$	1.25	1.0B	30.0	64.7M	4.14	3.60	7.74	3.9 ×
Flow-matching generative models, one-step								
iMF-B	100	89M	935	49.5M	0.12	3.77	3.89	240.4 ×
iMF-L	100	409M	1252	49.5M	0.51	3.76	4.27	293.2 ×
iMF-XL	100	610M	1489	49.5M	0.77	3.76	4.53	328.7 ×

Table 5: Reconstruction performance before and after FDDA. We compare the pretrained and FDDA-adapted decoders using PSNR, SSIM, rFID, and LPIPS. FDDA is trained on generation-time latents, and we report its effect on reconstruction performance here.

Models	space	#params	Pretrained Decoder				FDDA-adapted Decoder			
			PSNR(↑)	SSIM (↑)	rFID(↓)	LPIPS(↓)	PSNR(↑)	SSIM(↑)	rFID(↓)	LPIPS(↓)
LlamaGen-B	discrete	71.9M	22.53	55.3	2.20	0.138	21.73	52.9	2.82	0.154
LlamaGen-L	discrete	71.9M	22.53	55.3	2.20	0.138	21.93	54.0	2.55	0.150
GigaTok-S-S	discrete	136.1M	22.47	54.5	1.01	0.132	21.22	50.8	1.42	0.163
TiTok-L-32	discrete	641.1M	17.55	35.9	2.22	0.329	16.57	33.5	1.64	0.357
TiTok-B-64	discrete	204.8M	18.74	39.8	1.69	0.258	17.84	37.4	1.66	0.284
VAR-$d16$	discrete	108.9M	24.37	64.0	0.92	0.104	22.21	58.5	2.05	0.141
VAR-$d20$	discrete	108.9M	24.37	64.0	0.92	0.104	19.48	57.7	2.35	0.158
VAR-$d24$	discrete	108.9M	24.37	64.0	0.92	0.104	20.43	56.9	2.38	0.148
iMF-B	continuous	83.7M	27.61	72.5	0.75	0.068	26.65	69.7	0.92	0.081
iMF-L	continuous	83.7M	27.61	72.5	0.75	0.068	27.06	71.0	0.86	0.076
iMF-XL	continuous	83.7M	27.61	72.5	0.75	0.068	27.15	71.1	0.89	0.076

System-Level Comparison. We compare FDDA with a broader set of class-conditional ImageNet 256×256 generative models. For systems without FD post-training, we include LlamaGen-L [Sun+24], GigaTok-S-S [Xio+25], TiTok-L-32 [Yu+24], VAR-$d24$ and VAR-$d30$ [Tia+24], BAR-L [Yu+26], SiT-XL/2 [Ma+24], MAR-L and MAR-H [Li+24], FlowAR-H [Ren+25], MAR-L with DeTok [Yan+26b], REG [Wu+25], SiT-XL/2-REPA [Yu+25], LightningDiT [YYW25], DDT-XL [Wan+26], REPA-E [Len+25], RAE-XL [Zhe+26], Drift-L [Den+26], and iMF-XL [Gen+26]. We also compare matched LlamaGen-L, GigaTok-S-S, TiTok-L-32, VAR-$d24$, and VAR-$d30$ models after generator-side FD post-training and after further decoder-side post-training with FDDA. As shown in Table 3, FD post-training lowers gFDr^6 for all matched models, and applying FDDA afterward yields further reductions. For VAR-$d24$, gFDr^6 decreases from 6.79 without FD post-training to 2.65 after generator-side FD post-training and to **1.67** after FDDA. This is the lowest gFDr^6 among the systems compared in Table 3.

5.3 Computational Efficiency of FDDA

We next compare the end-to-end wall-clock time of decoder-side and generator-side FD post-training. As shown in Table 4, FDDA reduces post-training time across all evaluated models, including the cost of constructing the generation-time latent pool. For autoregressive models, the reduction ranges from 3.6-23.6×. The reduction is substantially larger for flow-based models, reaching 240.4-328.7× on iMF [Gen+26].

This efficiency comes from three factors. First, FDDA uses only one epoch of decoder adaptation,[1] whereas generator-side FD post-training uses 1.25 epochs for autoregressive models and 100 epochs for

[1]Following Yang et al. [Yan+26a], one epoch corresponds to training on 1.28 million generated samples; 1.25 epochs therefore correspond to training on 1.60 million generated samples.

iMF. Second, FDDA caches generation-time latents before training, so gradients pass only through the decoder. Generator-side optimization must additionally backpropagate through the latent generator and its sampling process, resulting in a deeper backward path. Third, FDDA updates fewer parameters because the decoder is smaller than the latent generator for most models, as shown in Table 4. These differences in training epochs, back-propagation depth, and parameter count account for the lower wall-clock cost of FDDA.

5.4 Additional Analyses

Effect of Training Epochs To examine whether longer decoder alignment yields further gains, we train FDDA for five epochs on GigaTok-S-S [Xio+25], both before and after generator-side FD post-training. As shown in Figure 2, gFDr^6 continues to decrease throughout all five epochs in both settings. Note that epoch-1 here is not directly comparable to the one-epoch models in Tables 1 and 2 due to different learning-rate schedules. Extending FDDA from one to five epochs reduces gFDr^6 from 6.59 to 5.10 before generator-side post-training and from 3.84 to 3.38 after it.

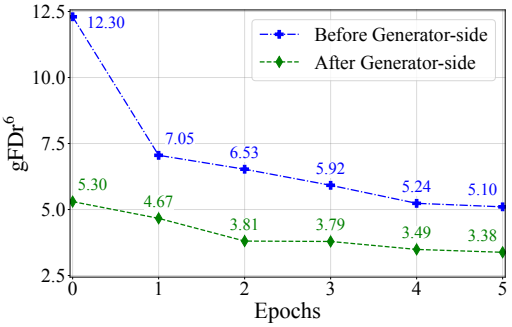


Figure 2: Effect of FDDA training epochs.

Reconstruction Performance. Table 5 compares reconstruction performance before and after FDDA using PSNR, SSIM, rFID, and LPIPS. Reconstruction quality generally decreases after decoder adaptation: PSNR and SSIM are lower across all models, while LPIPS increases in all cases; rFID also worsens for most models, although it improves slightly for the TiTok models. This follows from the training objective of FDDA: the decoder is adapted on generation-time latents rather than reconstruction latents. This suggests a trade-off between reconstruction fidelity and adaptation to the generation-time latent distribution. Appendix C ablates the choice of decoder-side FD post-training latents, showing that generation-time latents are key to the observed gains. Qualitative comparisons are provided in Appendix D.

6 Conclusion

In this work, we extend Fréchet distributional post-training from the generator to the decoder in latent generative models. Motivated by the decoder train-generation mismatch, where a decoder trained on reconstruction latents is ultimately applied to generator-produced latents, we introduce **Fréchet Distributional Decoder Alignment (FDDA)**. FDDA keeps the pretrained latent generator fixed and directly optimizes the decoder on generation-time latents using an FD objective. Across diverse flow-based and autoregressive latent generative models, a single epoch of FDDA consistently improves generation quality from the original pretrained checkpoints while requiring substantially less post-training cost than generator-side FD optimization. FDDA further improve gFDr^6 after the generator has already undergone FD post-training, showing that generator-side optimization does not exhaust the benefits of distribution-level post-training. More broadly, our results show that distribution-level post-training need not be confined to the latent generator. Generation quality depends both on the generated latent distribution and on how the decoder maps it to images. By aligning the decoder with generation-time latents, FDDA provides a simple and efficient complement to generator-side post-training.

References

[Bla+24] K. Black, M. Janner, Y. Du, I. Kostrikov, and S. Levine. “Training diffusion models with reinforcement learning”. In: *ICLR*. 2024 (cit. on pp. 1, 3).

[Cla+24] K. Clark, P. Vicol, K. Swersky, and D. J. Fleet. “Directly fine-tuning diffusion models on differentiable rewards”. In: *ICLR*. 2024 (cit. on p. 3).

[Den+24] F. Deng, Q. Wang, W. Wei, M. Grundmann, and T. Hou. “Prdp: proximal reward difference prediction for large-scale reward finetuning of diffusion models”. In: *CVPR*. 2024 (cit. on p. 3).

[Den+26] M. Deng, H. Li, T. Li, Y. Du, and K. He. “Generative modeling via drifting”. *ArXiv* (2026) (cit. on p. 9).

[Doa+20] K. D. Doan, S. Manchanda, F. Wang, S. S. Keerthi, A. Bhowmik, and C. K. Reddy. “Image generation via minimizing fréchet distance in discriminator feature space”. *ArXiv* (2020) (cit. on p. 3).

[ERO21] P. Esser, R. Rombach, and B. Ommer. “Taming transformers for high-resolution image synthesis”. In: *CVPR*. 2021 (cit. on pp. 1–4).

[Fan+23] Y. Fan, O. Watkins, Y. Du, H. Liu, M. Ryu, C. Boutilier, P. Abbeel, M. Ghavamzadeh, K. Lee, and K. Lee. “Dpok: reinforcement learning for fine-tuning text-to-image diffusion models”. In: *NeurIPS*. 2023 (cit. on p. 3).

[Fan+26] X. Fang, W. Shu, T. Xu, W. Mou, D. Kong, and T. G. J. Rudner. “Bridging reconstruction and generation: a latent distribution perspective on evaluation and improvement”. *ArXiv* (2026) (cit. on pp. 1–5).

[Gao+26] M. Gao, J. Zhou, K. Gai, C. Yu, and H. Tang. “Advfd: boosting visual generation via adversarial fréchet distance loss”. *ArXiv* (2026) (cit. on pp. 1, 3, 4).

[Gen+25] Z. Geng, M. Deng, X. Bai, J. Z. Kolter, and K. He. “Mean flows for one-step generative modeling”. In: *NeurIPS*. 2025 (cit. on p. 4).

[Gen+26] Z. Geng, Y. Lu, Z. Wu, E. Shechtman, J. Z. Kolter, and K. He. “Improved mean flows: on the challenges of fastforward generative models”. In: *CVPR*. 2026 (cit. on pp. 3, 4, 6, 9, 14, 15).

[He+22] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. “Masked autoencoders are scalable vision learners”. In: *CVPR*. 2022 (cit. on pp. 5, 6, 14).

[Heu+17] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. “Gans trained by a two time-scale update rule converge to a local nash equilibrium”. In: *NeurIPS*. 2017 (cit. on pp. 3, 4, 6).

[Len+25] X. Leng, J. Singh, Y. Hou, Z. Xing, S. Xie, and L. Zheng. “Repa-e: unlocking vae for end-to-end tuning with latent diffusion transformers”. In: *ICCV*. 2025 (cit. on p. 9).

[Li+24] T. Li, Y. Tian, H. Li, M. Deng, and K. He. “Autoregressive image generation without vector quantization”. In: *NeurIPS*. 2024 (cit. on p. 9).

[Lin+26] S. Lin, C. Yang, Z. Lin, H. Chen, and H. Fan. “Continuous adversarial flow models”. In: *ECCV*. 2026 (cit. on pp. 1, 3).

[Luo+25] W. Luo, C. Zhang, D. Zhang, and Z. Geng. “David and goliath: small one-step model beats large diffusion with score post-training”. In: *ICML*. 2025 (cit. on p. 1).

[Ma+24] N. Ma, M. Goldstein, M. S. Albergo, N. M. Boffi, E. Vanden-Eijnden, and S. Xie. “Sit: exploring flow and diffusion-based generative models with scalable interpolant transformers”. In: *ECCV*. 2024 (cit. on p. 9).

[MH21] A. Mathiasen and F. Hvilshoj. “Backpropagating through fréchet inception distance”. *ArXiv* (2021) (cit. on p. 3).

[OVK17] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. “Neural discrete representation learning”. In: *NeurIPS*. 2017 (cit. on pp. 1–4).

[Oqu+24] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. “Dinov2: learning robust visual features without supervision”. *TMLR* (2024) (cit. on pp. 5, 6, 14).

[PX23] W. S. Peebles and S. Xie. “Scalable diffusion models with transformers”. In: *ICCV*. 2023 (cit. on p. 4).

[Qiu+25] K. Qiu, X. Li, H. Chen, J. Kuen, X. Xu, J. Gu, Y. Luo, B. Raj, Z. Lin, and M. Savvides. “Image tokenizer needs post-training”. *ArXiv* (2025) (cit. on pp. 2–5).

[Rad+21] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. “Learning transferable visual models from natural language supervision”. In: *ICML*. 2021 (cit. on pp. 5, 6, 14).

[Ren+25] S. Ren, Q. Yu, J. He, X. Shen, A. L. Yuille, and L.-C. Chen. “Flowar: scale-wise autoregressive image generation meets flow matching”. In: *ICML*. 2025 (cit. on p. 9).

[Rom+22] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. “High-resolution image synthesis with latent diffusion models”. In: *CVPR*. 2022 (cit. on pp. 1–4).

[Sal+16] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. “Improved techniques for training gans”. In: *NeurIPS*. 2016 (cit. on p. 6).

[Sun+24] P. Sun, Y. Jiang, S. Chen, S. Zhang, B. Peng, P. Luo, and Z. Yuan. “Autoregressive model beats diffusion: llama for scalable image generation”. *ArXiv* (2024) (cit. on pp. 1, 3, 4, 6, 9, 15).

[Sze+16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. “Rethinking the inception architecture for computer vision”. In: *CVPR*. 2016 (cit. on pp. 4, 5, 14).

[Tia+24] K. Tian, Y. Jiang, Z. Yuan, B. Peng, and L. Wang. “Visual autoregressive modeling: scalable image generation via next-scale prediction”. In: *NeurIPS*. 2024 (cit. on pp. 3, 4, 6, 9, 15).

[Tsc+25] M. Tschannen, A. Gritsenko, X. Wang, M. F. Naeem, I. Alabdulmohsin, N. Parthasarathy, T. Evans, L. Beyer, Y. Xia, B. Mustafa, et al. “Siglip 2: multilingual vision-language encoders with improved semantic understanding, localization, and dense features”. *Arxiv* (2025) (cit. on pp. 5, 6, 14).

[Wal+24] B. Wallace, M.-H. Dang, R. Rafailov, L. Zhou, A. Lou, S. Purushwalkam, S. Ermon, C. Xiong, S. R. Joty, and N. Naik. “Diffusion model alignment using direct preference optimization”. In: *CVPR*. 2024 (cit. on p. 3).

[Wan+26] S. Wang, Z. Tian, W. Huang, and L. Wang. “Ddt: decoupled diffusion transformer”. In: *CVPR*. 2026 (cit. on p. 9).

[Woo+23] S. Woo, S. Debnath, R. Hu, X. Chen, Z. Liu, I.-S. Kweon, and S. Xie. “Convnext v2: co-designing and scaling convnets with masked autoencoders”. In: *CVPR*. 2023 (cit. on pp. 5, 6, 14).

[Wu+25] G. Wu, S. Zhang, R. Shi, S. Gao, Z. Chen, L. Wang, Z. Chen, H. Gao, Y. Tang, J. Yang, M.-M. Cheng, and X. Li. “Representation entanglement for generation:training diffusion transformers is much easier than you think”. In: *NeurIPS*. 2025 (cit. on p. 9).

[Xio+25] T. Xiong, J. H. Liew, Z. Huang, J. Feng, and X. Liu. “Gigatok: scaling visual tokenizers to 3 billion parameters for autoregressive image generation”. In: *ICCV*. 2025 (cit. on pp. 3, 4, 6, 9, 10, 14, 15).

[Xu+23] J. Xu, X. Liu, Y. Wu, Y. Tong, Q. Li, M. Ding, J. Tang, and Y. Dong. “Imagereward: learning and evaluating human preferences for text-to-image generation”. In: *NeurIPS*. 2023 (cit. on pp. 1, 3).

[Yan+26a] J. Yang, Z. Geng, X. Ju, Y. Tian, and Y. Wang. “Representation fréchet loss for visual generation”. *ArXiv* (2026) (cit. on pp. 1, 3–9, 14).

[Yan+26b] J. Yang, T. Li, L. Fan, Y. Tian, and Y. Wang. “Latent denoising makes good tokenizers”. In: *ICLR*. 2026 (cit. on p. 9).

[Yan+24] K. Yang, J. Tao, J. Lyu, C. Ge, J. Chen, Q. Li, W. Shen, X. Zhu, and X. Li. “Using human feedback to fine-tune diffusion models without any reward model”. In: *CVPR*. 2024 (cit. on p. 3).

[YYW25] J. Yao, B. Yang, and X. Wang. “Reconstruction vs. generation: taming optimization dilemma in latent diffusion models”. In: *CVPR*. 2025 (cit. on p. 9).

[Yu+26] Q. Yu, Q. Liu, J. He, X. Zhang, Y. Liu, L.-C. Chen, and X. Chen. “Autoregressive image generation with masked bit modeling”. In: *ICML*. 2026 (cit. on p. 9).

[Yu+24] Q. Yu, M. Weber, X. Deng, X. Shen, D. Cremers, and L.-C. Chen. “An image is worth 32 tokens for reconstruction and generation”. In: *NeurIPS*. 2024 (cit. on pp. 4, 6, 9, 15).

[Yu+25] S. Yu, S. Kwak, H. Jang, J. Jeong, J. Huang, J. Shin, and S. Xie. “Representation alignment for generation: training diffusion transformers is easier than you think”. In: *ICLR*. 2025 (cit. on p. 9).

[Zha+26] J. Zhang, Y. Lin, W. Long, and S. Gu. “Beyond token-level cross-entropy: fréchet distributional post-training for autoregressive image generation”. *ArXiv* (2026) (cit. on pp. 1, 3, 4, 6).

[Zha+18] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. “The unreasonable effectiveness of deep features as a perceptual metric”. In: *CVPR*. 2018 (cit. on p. 6).

[Zhe+26] B. Zheng, N. Ma, S. Tong, and S. Xie. “Diffusion transformers with representation autoencoders”. In: *ICLR*. 2026 (cit. on p. 9).

Appendix

A Implementation and Training Details

Generation-Time Latent Pools. For each generator checkpoint, we sample and cache 1.28M generation-time latents, balanced across the $1{,}000$ ImageNet classes. These cached latents form the training set for decoder-side FD post-training, so one FDDA epoch corresponds to 1.28M decoder inputs. For autoregressive models, generated tokens are converted to generation-time latents through the frozen tokenizer codebook before being cached as decoder inputs. For flow-based models, we directly cache the final latent states produced by the generator. In both cases, latent pools are constructed using the checkpoint-specific classifier-free guidance (CFG) and sampling configurations used for evaluation.

Decoder-Side FD Post-Training. All FDDA models are trained for one epoch using AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight decay 10^{-4}. We use a batch size of 32, corresponding to $40{,}000$ optimization steps per epoch. The learning rate is linearly warmed up to 10^{-5} over the first 500 steps and then decayed to 10^{-6} using a cosine schedule over the remaining training steps. We apply gradient clipping with maximum norm 1.0 and use `bf16` autocast. All experiments are run on a single NVIDIA H100 GPU with 80GB memory. For the longer-training analysis in Figure 2, we independently train GigaTok-S-S [Xio+25] for five epochs using the same optimization hyperparameters and learning-rate schedule. The only change is the total training horizon, from $40{,}000$ to $200{,}000$ steps, so the cosine decay is correspondingly stretched over the longer run.

Reference Statistics. We use the ImageNet reference statistics released by Yang et al. [Yan+26a] on Hugging Face for six feature extractors: Inception-v3 [Sze+16], ConvNeXt-v2 [Woo+23], DINOv2 [Oqu+24], MAE [He+22], SigLIP2 [Tsc+25], and CLIP [Rad+21]. The statistics for Inception-v3, MAE, and SigLIP2 are used during FDDA training, while all six are used for evaluation.

B Limitations

Our study has several limitations. First, although we evaluate FDDA across 11 configurations from autoregressive and flow-based latent generative models, all experiments use class-conditional ImageNet generation at 256×256 resolution. We have not yet tested FDDA for text-to-image generation, higher-resolution generation, or other data modalities.

Second, FDDA is designed to improve generation by adapting the decoder to generation-time latents. As shown in Table 5, reconstruction PSNR and SSIM decrease after adaptation across the evaluated models, while LPIPS increases and rFID worsens in most cases. This suggests a trade-off between reconstruction fidelity and adaptation to the latent distribution encountered during generation. For applications that require both reconstruction and generation, an additional reconstruction-preserving objective may be needed during decoder alignment.

Third, FDDA adapts the decoder to a fixed generation-time latent distribution $p_g(z)$ produced by a particular generator checkpoint and inference configuration. Changing the generator or its sampling configuration may change this distribution, in which case the decoder may need to be adapted again.

Finally, FDDA optimizes Fréchet distances in a finite set of pretrained representation spaces. Although we additionally evaluate with three representations held out from FDDA training, Fréchet objectives capture only first- and second-order feature statistics and may not reflect all aspects of perceptual or semantic generation quality.

C Effect of Decoder Post-Training Latents

FDDA performs decoder-side FD post-training on generation-time latents z_g. To examine the role of this choice, we replace z_g with reconstruction latents z_r obtained by encoding real images, while keeping the decoder-side FD post-training procedure otherwise unchange. We compare the two latent choices on iMF-B, iMF-L, and iMF-XL [Gen+26], starting from either the original pretrained models or their generator-side FD-post-trained counterparts. In all cases, we evaluate generation using samples from the corresponding generator.

Table 6: Effect of decoder post-training latents on iMF. We compare decoder-side FD post-training using reconstruction latents z_r or generation-time latents z_g, starting from either the original pretrained generator or its FD-post-trained counterpart. ✗and ✔indicate the absence and presence of FD post-training, respectively, for the component named in each column. All reported metrics evaluate generated images. The percentage beside gFDr^6 is the relative change from the row without decoder FD in the same generator setting; downward arrows indicate improvement.

Models	Generator FD	Decoder FD	Decoder latents	gFID (↓)	gIS (↑)	gFD_{conv} (↓)	gFD_{dino} (↓)	gFD_{mae} (↓)	gFD_{sig} (↓)	gFD_{clip} (↓)	gFDr^6 (↓)
iMF-B	✗	✗	-	3.49	253.7	188.08	206.17	0.487	19.65	156.14	15.40
	✗	✔	z_r	3.27	265.1	229.78	172.91	0.361	15.13	141.03	12.91 ↓16.2%
	✗	✔	z_g	1.65	268.1	115.47	144.39	0.232	8.10	105.06	8.54 ↓44.5%
	✔	✗	-	**0.86**	**307.1**	65.97	89.05	0.154	5.46	70.35	5.57
	✔	✔	z_r	1.08	305.5	113.59	84.02	0.158	5.29	68.01	5.58 ↑0.2%
	✔	✔	z_g	0.91	304.0	74.66	80.48	0.100	3.91	59.69	4.54 ↓18.5%
iMF-L	✗	✗	-	1.88	275.4	105.53	111.56	0.282	11.30	101.18	9.11
	✗	✔	z_r	1.79	285.2	157.99	92.29	0.187	7.48	85.72	7.13 ↓21.7%
	✗	✔	z_g	1.26	281.0	75.93	83.77	0.137	5.12	72.12	5.48 ↓39.8%
	✔	✗	-	**0.77**	**303.4**	**43.60**	38.89	0.050	2.46	38.85	2.72
	✔	✔	z_r	0.98	301.6	101.26	38.36	0.089	2.92	39.21	3.19 ↑17.3%
	✔	✔	z_g	0.80	299.1	65.13	36.92	0.035	2.05	33.72	2.43 ↓10.7%
iMF-XL	✗	✗	-	1.81	281.9	97.57	103.01	0.261	10.37	94.58	8.44
	✗	✔	z_r	1.67	289.0	141.05	85.23	0.159	6.56	76.87	6.35 ↓24.8%
	✗	✔	z_g	1.14	288.3	67.12	77.30	0.117	4.67	67.74	5.03 ↓40.4%
	✔	✗	-	**0.76**	**306.7**	**40.34**	33.85	0.043	2.21	35.74	2.46
	✔	✔	z_r	0.95	299.2	90.05	34.10	0.070	2.41	34.36	2.74 ↑11.4%
	✔	✔	z_g	0.78	303.7	57.53	32.51	0.030	1.82	31.61	2.21 ↓10.2%

Table 6 shows that both latent choices reduce gFDr^6 relative to the original pretrained models, but training on z_g produces larger gains. With z_r, gFDr^6 falls from 15.40 to 12.91 on iMF-B, from 9.11 to 7.13 on iMF-L, and from 8.44 to 6.35 on iMF-XL. The corresponding results with z_g are 8.54, 5.48, and 5.03. Training on z_g also gives lower gFID and lower FD in each of the five other reported feature spaces than training on z_r. Notably, gFD_{conv} increases under z_r adaptation for all three models, whereas it decreases under z_g adaptation.

After generator-side FD post-training, the difference becomes more pronounced in gFDr^6. Training on z_r changes it from 5.57 to 5.58 on iMF-B, from 2.72 to 3.19 on iMF-L, and from 2.46 to 2.74 on iMF-XL. Training on z_g instead lowers these values to 4.54, 2.43, and 2.21. It also yields lower FD than z_r adaptation in all five other reported feature spaces for each model. The gains from FDDA therefore depend on the decoder post-training inputs: after generator-side FD post-training, using z_r reverses the gFDr^6 benefit, while using the latents encountered at generation time retains it.

D Qualitative comparisons

We present qualitative comparisons under three settings. First, we evaluate FDDA before generator-side FD post-training. On TiTok-L-32 [Yu+24], Figure 3 compares the original pretrained model with its FDDA-adapted counterpart, keeping the latent generator fixed. This comparison isolates the effect of decoder-side adaptation.

Second, we evaluate FDDA after generator-side FD post-training. Figure 4 compares the generator-side FD post-trained TiTok-L-32 model before and after applying FDDA, showing the additional changes introduced by decoder adaptation when the latent generator has already been post-trained.

Finally, we compare the original pretrained models with models after both generator-side FD post-training and FDDA. For TiTok-L-32, Figures 5-13 show examples from 10 ImageNet classes. We also include results on LlamaGen-L [Sun+24], GigaTok-S-S [Xio+25], VAR-$d24$ [Tia+24], and iMF-XL [Gen+26] in Figures 14-17. Across these examples, the adapted models generally show slightly richer fine-grained details, fewer visible artifacts, and more semantically consistent generations. These visual differences are consistent with the improvements observed in the quantitative evaluation.



Figure 3: Generated samples before generator-side FD post-training. Top: original pretrained model. Bottom: after FDDA.



Figure 4: Generated samples after generator-side FD post-training. Top: generator-side FD post-training. Bottom: after further applying FDDA.



Figure 5: Class 14: indigo bunting. Top: original pretrained model. Bottom: generator-side FD post-training + FDDA.



Figure 6: Class 22: bald eagle. Top: original pretrained model. Bottom: generator-side FD post-training + FDDA.



Figure 7: Class 42: agama. Top: original pretrained model. Bottom: generator-side FD post-training + FDDA.



Figure 8: Class 81: ptarmigan. Top: original pretrained model. Bottom: generator-side FD post-training + FDDA.



Figure 9: Class 88: macaw. Top: original pretrained model. Bottom: generator-side FD post-training + FDDA.



Figure 10: Class 108: sea anemone. Top: original pretrained model. Bottom: generator-side FD post-training + FDDA.



Figure 11: Class 130: flamingo. Top: original pretrained model. Bottom: generator-side FD post-training + FDDA.

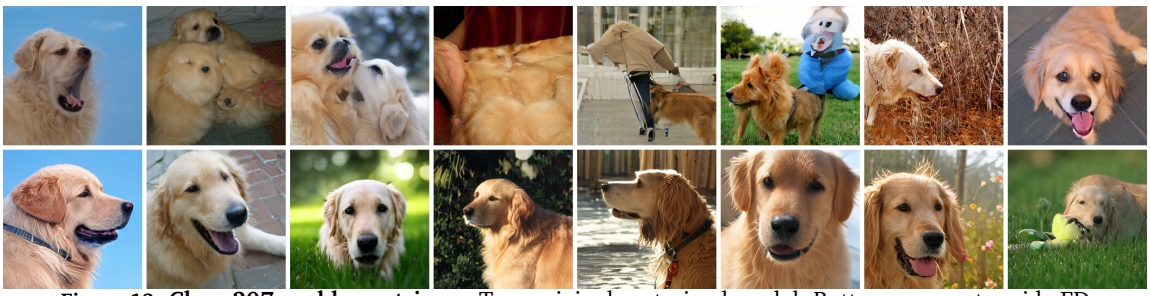


Figure 12: Class 207: golden retriever. Top: original pretrained model. Bottom: generator-side FD post-training + FDDA.



Figure 13: Class 483: castle. Top: original pretrained model. Bottom: generator-side FD post-training + FDDA.



Figure 14: Generated samples from LlamaGen-L. Top: original pretrained model. Bottom: generator-side FD post-training + FDDA.



Figure 15: Generated samples from GigaTok-S-S. Top: original pretrained model. Bottom: generator-side FD post-training + FDDA.



Figure 16: Generated samples from VAR-$d24$. Top: original pretrained model. Bottom: generator-side FD post-training + FDDA.



Figure 17: Generated samples from iMF-XL. Top: original pretrained model. Bottom: generator-side FD post-training + FDDA.