LATENT PRIOR GANS: EFFICIENT DECODER REFINEMENT VIA INHERITED GENERATIVE STRUCTURE

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern latent generative models learn strong generative structure through stable, non-adversarial training objectives in their latent generators. Their success changes a key premise of adversarial generation: this structure need no longer be learned from scratch through adversarial optimization, but can instead be inherited from a pretrained model. We refer to this inherited generative structure, comprising the learned generation-time latent distribution and pretrained latent-to-image decoding knowledge, as the latent prior. We introduce Latent Prior GANs (LP-GANs), which repurpose adversarial learning from discovering generative structure from scratch to targeted generation-time decoder refinement. LP-GANs freezes the pretrained latent generator and adversarially optimizes its decoder on generation-time latents, directly correcting the decoder train-generation mismatch: conventional decoders are optimized on reconstruction latents but deployed on generated latents. The reconstruction and perceptual losses require paired images, which generated latents do not provide, so we construct intermediate latents that still correspond to a real image. Across four latent generators and their model scales, five epochs of adversarial adaptation are enough to improve generation quality over the pretrained model. We hope this makes adversarial training cheaper when strong pretrained models are already available.

1 INTRODUCTION

Generative adversarial networks (GANs) directly optimize generated samples against the data distribution and have been highly effective for high-fidelity image synthesis (Goodfellow et al., 2014; Brock et al., 2018; Karras et al., 2019; 2020; 2021). However, GANs are also notoriously difficult and costly to train. In large part, this is because a from-scratch GAN is asked to do two things at once: find the structure that makes synthesis possible, and turn that structure into high-quality samples. In practice, both tasks fall to the same adversarial process, which is why large-scale GAN training is difficult to optimize (Figure 1(a)).

Rather than learning such structure from scratch, one can start from generative structure learned beforehand, as already offered by pretrained latent generative models. We therefore call their inherited structure the *latent prior*. It contains what the latent generator has learned about its generation-time latent distribution, together with the latent-to-image decoding knowledge in the pretrained decoder (Esser et al., 2021; Rombach et al., 2022; Peebles & Xie, 2023; Geng et al., 2025; 2026). Adversarial training can then start from this prior, instead of paying the full cost of discovering such generative structure from scratch.

However, this latent prior contains an internal mismatch: its two components were never trained to match each other. The two are learned in separate stages (Esser et al., 2021; Rombach et al., 2022). During tokenizer training, the decoder sees reconstruction latents $z_r \sim p_r(z)$ from the encoder. At generation time the latents instead come from the generator, $z_g \sim p_g(z)$. The generator models p_r only approximately, so $p_g \neq p_r$ in general and the decoder is applied off the distribution under which it was optimized. We refer to this discrepancy as the *decoder train-generation mismatch* (Fang et al., 2026) (Figure 1(b)). And p_g is already the distribution used at deployment, so the mismatch left is at the decoder, which has never been optimized on it.

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.

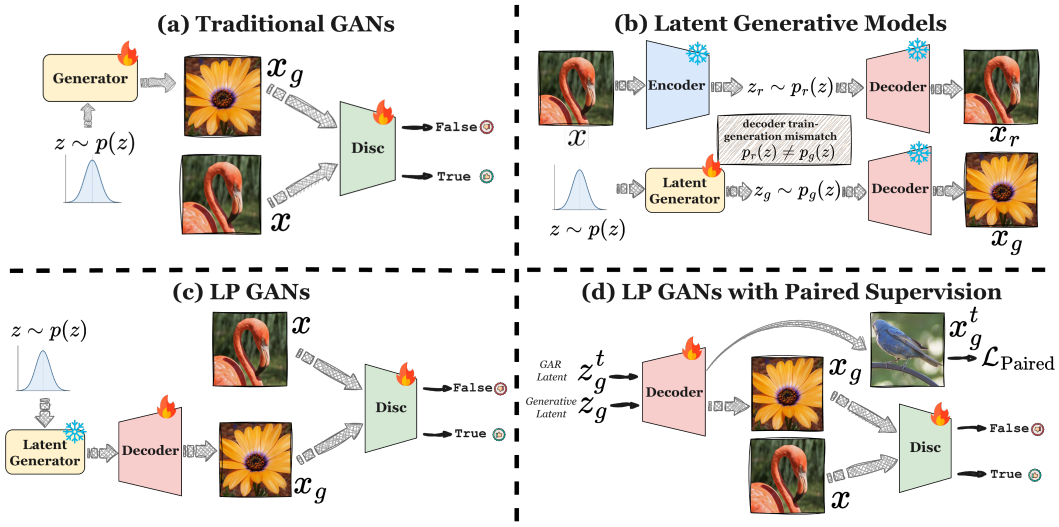

Figure 1: **Overview and motivation of Latent Prior GANs (LP-GANs).** **(a)** Conventional GANs couple generative structure discovery and refinement within a training-intensive adversarial process. **(b)** Pretrained latent generative models provide learned generative structure, but their decoders are optimized on reconstruction latents and deployed on generated latents, creating a decoder train-generation mismatch. **(c)** LP-GANs inherit this latent prior, freeze the latent generator, and correct this mismatch through targeted, generation-time decoder refinement. **(d)** Generation-aware intermediate latents provide paired supervision that anchors decoder adaptation toward the generation-time latent distribution.

This raises a natural question:

> *Can adversarial optimization be repurposed from learning generative structure from scratch to refining a pretrained decoder for generation-time latents?*

We answer this question with **Latent Prior GANs (LP-GANs)**, shown in Figure 1(c). The latent generator already defines $p_g(z)$, the distribution encountered at generation time, so we keep it fixed and place adversarial optimization on the decoder alone. The decoder is trained on latents drawn from $p_g(z)$, which changes how that distribution is decoded rather than the distribution itself.

Fully generated latents differ from reconstruction latents in one important way: they have no paired ground-truth images, so the correspondence needed for perceptual supervision is missing. We therefore use intermediate latents along a trajectory from reconstruction to generation (Fang et al., 2026): they move progressively toward generation-time latents while remaining paired with their source images. This *generation-aware paired supervision* anchors the decoder through paired reconstruction and perceptual objectives (Johnson et al., 2016; Zhang et al., 2018), while adversarial learning optimizes it on fully generated latents (Figure 1(d)).

Across four latent generators and their scales, five epochs of adversarial adaptation are enough to improve generation quality, at a small fraction of the cost of learning structure from scratch.

Our main contributions are:

1. We introduce **Latent Prior GANs (LP-GANs)**, which inherit generative structure from pretrained latent models and repurpose adversarial learning from generative structure discovery to targeted decoder refinement.
2. We directly optimize the decoder under the fully generated latent distribution $p_g(z)$ while keeping the latent generator frozen, and use generation-aware intermediate latents to provide paired reconstruction and perceptual supervision that anchors this refinement.
3. Across four latent generators and their scales, LP-GANs improve generation quality within five epochs of adversarial adaptation, which makes decoder refinement over inherited structure both effective and cheap.

2 Related Work

Adversarial image generation. Generative adversarial networks, or GANs (Goodfellow et al., 2014), learn the image distribution by pitting two networks against each other. Later work made them easier to train and improved their samples, by changing the objective (Arjovsky et al., 2017; Gulrajani et al., 2017), regularizing the discriminator (Miyato et al., 2018; Mescheder et al., 2018), or scaling up the architecture (Brock et al., 2018; Karras et al., 2019; 2020; 2021). All of these GANs start from random noise. A single adversarial process discovers the generative structure required for synthesis and refines it into high-quality samples. LP-GANs removes the first of these responsibilities: generative structure is inherited as a latent prior from a pretrained latent generative model, leaving adversarial optimization to focus on refinement.

Adversarial post-training of pretrained generators. Recent work applies adversarial post-training, or adversarial optimization of a pretrained generative model. One line uses adversarial or distribution-matching objectives primarily to compress sampling into few or single steps: ADD (Sauer et al., 2023), LADD (Sauer et al., 2024), SiDA (Zhou et al., 2025b), and SDXL-Lightning (Lin et al., 2024) incorporate adversarial losses into diffusion distillation, while UFO-Gen (Xu et al., 2024) initializes from a pretrained diffusion model to train a one-step diffusion-GAN hybrid and D2O (Zheng & Yang, 2025) treats diffusion training as generative pre-training followed by lightweight GAN fine-tuning for one-step generation. Beyond distillation, ADT (Shen et al., 2025) adversarially fine-tunes a pretrained generator to better align inference-time outputs with real data. These methods differ from LP-GANs in where subsequent optimization is applied. They modify, distill, or fine-tune the generative model or its sampling process, whereas LP-GANs keeps the latent generator entirely frozen and uses it only as a source of generation-time latents $z_g \sim p_g(z)$. The adversarial signal acts exclusively on the downstream decoder. LP-GANs thus aims not at faster sampling, but at aligning the decoder with a latent distribution the generator already produces.

Latent decoding and generation-time alignment. Latent generative models build the generative distribution over a learned representation, or latent space. The representation is discrete in VQ-VAE, VQ-VAE-2, and VQGAN (van den Oord et al., 2017; Razavi et al., 2019; Esser et al., 2021), and continuous in latent diffusion and flow-based models (Rombach et al., 2022; Peebles & Xie, 2023; Geng et al., 2025; 2026). Decoders in these pipelines are commonly trained with reconstruction, perceptual, and adversarial objectives (Esser et al., 2021; Rombach et al., 2022), but are typically optimized on reconstruction latents $z_r \sim p_r(z)$ and deployed on generated latents, giving rise to the decoder train-generation mismatch (Fang et al., 2026).

Recent work narrows this gap by exposing the decoder to latents that move toward generation while preserving correspondence with source images. RobusTok (Qiu et al., 2025) performs generator-aware decoder post-training, introducing for autoregressive generators a preservation ratio that retains a fraction of reconstruction tokens during sampling. Generation-aware reconstruction (Fang et al., 2026) constructs a reconstruction-to-generation trajectory and treats the resulting intermediate latents as a continuum from $p_r(z)$ toward $p_g(z)$ under paired supervision. In both cases, supervision stops short of fully generated latents from $p_g(z)$.

LP-GANs extends this generation-aware perspective to the generation-time distribution itself: it optimizes the decoder directly on fully generated latents, which admit no paired reconstruction target, and uses intermediate reconstruction-to-generation latents only to anchor this adaptation. The key distinction is therefore not adversarial decoder training, already standard in these pipelines, but the latent distribution under which it is applied.

3 Preliminaries: Latent Generative Models

Two-stage latent generative modeling. We focus on diffusion- and flow-based latent generative models with continuous latent spaces (Rombach et al., 2022; Peebles & Xie, 2023; Geng et al., 2025; 2026). These models typically adopt a two-stage formulation. The first stage learns a tokenizer, consisting of an encoder $\mathcal{E}_\theta$ and a decoder $\mathcal{D}_\phi$, which map an image $x \sim p_{\text{data}}$ to a latent representation $z_r = \mathcal{E}_\theta(x)$ and back to an image $\hat{x} = \mathcal{D}_\phi(z_r)$. The tokenizer is optimized with reconstruction and perceptual objectives (Johnson et al., 2016; Zhang et al., 2018), commonly together with an adversarial objective (Goodfellow et al., 2014):

$$\min_{\theta,\,\phi}\ \mathbb{E}_{x\sim p_{\text{data}}}\Big[\underbrace{\mathcal{L}_{\text{rec}}(x,\hat{x}) + \mathcal{L}_{\text{perc}}(x,\hat{x})}_{\text{paired objectives}} + \underbrace{\lambda\,\mathcal{L}_{\text{adv}}(\hat{x})}_{\text{distributional objective}}\Big], \tag{1}$$

where λ balances the adversarial term against the paired terms, and $\hat{x} = \mathcal{D}_\phi(\mathcal{E}_\theta(x))$: in every term of Equation 1, the decoder receives only encoder-produced latents. The first two terms are *paired*: they compare the reconstruction $\hat{x}$ with its corresponding source image x, either in pixel or feature space. The adversarial term is *distributional* and requires no such image-level correspondence. The second stage then learns a latent generator $\mathcal{G}_\varphi$ over the latents produced by the frozen encoder, so that samples drawn from $\mathcal{G}_\varphi$ can be decoded by $\mathcal{D}_\phi$ into images. The discussion below does not depend on how $\mathcal{G}_\varphi$ is parameterized; it only needs access to its generation-time sampling procedure.

Reconstruction and generation latents. The two-stage latent generative pipeline exposes the decoder to distinct reconstruction and generation latent distributions. During first-stage training, the decoder receives *reconstruction latents* produced by the encoder,

$$z_r \sim p_r(z), \qquad p_r = (\mathcal{E}_\theta)_{\#}\, p_{\text{data}},$$

where p_r is the pushforward of the data distribution through the encoder. In the second stage, the tokenizer is frozen and a latent generator $\mathcal{G}_\varphi$ is trained to approximately model p_r. At generation time, the decoder instead receives *generation latents* produced by the trained latent generator through its deployed sampling procedure,

$$z_g = \mathcal{S}(\mathcal{G}_\varphi, \epsilon, \xi) \sim p_g(z), \qquad \epsilon \sim \mathcal{N}(0, I), \quad \xi \sim p_\xi,$$

where ϵ denotes the noise initialization, ξ denotes any additional randomness used by the sampler, and p_g is the latent distribution induced by the deployed generation procedure. Although $\mathcal{G}_\varphi$ is trained to model p_r, the resulting generation-time distribution need not match it exactly, so $p_g \neq p_r$ in general (Fang et al., 2026).

Decoder train-generation mismatch. In Equation 1, all updates to the decoder are induced by reconstruction latents $z_r \sim p_r$, whereas at generation time the same decoder is applied to generation latents $z_g \sim p_g$. When $p_g \neq p_r$, this creates a *decoder train-generation mismatch* (Fang et al., 2026): the decoder is deployed on a latent distribution different from the one on which it was optimized. This mismatch cannot be addressed by simply replacing z_r with z_g in the first-stage objective, because a generated latent z_g has no corresponding source image x, leaving the paired reconstruction and perceptual objectives undefined. The adversarial objective, by contrast, remains directly applicable under p_g because it requires no image-level correspondence. Together, the decoder train-generation mismatch and this supervision asymmetry motivate LP-GANs, which apply adversarial supervision directly to generation-time latents $z_g \sim p_g$. We introduce LP-GANs next.

4 Latent Prior GANs

We introduce **Latent Prior GANs (LP-GANs)**, which inherit the *latent prior* provided by pretrained latent generative models and correct the decoder train-generation mismatch identified in Section 3. LP-GANs directly applies adversarial supervision to the decoder under the fully generated latent distribution $p_g(z)$ while keeping the latent generator frozen (Section 4.1). Since fully generated latents admit no paired target, we use generation-aware intermediate latents to provide paired supervision that anchors this generation-time refinement (Section 4.2).

4.1 Generation-Time Decoder Refinement

LP-GANs optimizes the decoder on the latent distribution it encounters at deployment. Starting from a pretrained latent generator $\mathcal{G}_\varphi$ and decoder $\mathcal{D}_{\phi_0}$, we keep $\mathcal{G}_\varphi$ frozen and initialize $\mathcal{D}_\phi$ from $\mathcal{D}_{\phi_0}$. We sample $z_g = \mathcal{S}(\mathcal{G}_\varphi, \xi) \sim p_g(z)$ with $\xi \sim p_\xi$, decode it as $\mathcal{D}_\phi(z_g)$, and let an image-space discriminator $\mathcal{C}_\omega$ distinguish the decoded samples from real images $x \sim p_{\text{data}}$:

$$\mathcal{L}^g_{\text{adv}}(\phi) = \mathbb{E}_{z_g \sim p_g}\Big[\mathcal{L}_{\text{adv}}\big(\mathcal{D}_\phi(z_g); \mathcal{C}_\omega\big)\Big]. \tag{2}$$

We adopt the DINO-based discriminator and adversarial training setup of Tian et al. (2024), as used in prior tokenizer training works (Chen et al., 2025b; Li et al., 2025). Throughout adaptation, $\mathcal{G}_\varphi$ and $\mathcal{E}_\theta$ remain fixed; only $\mathcal{D}_\phi$ and $\mathcal{C}_\omega$ are updated. Consequently, LP-GANs leaves the generation-time latent distribution p_g unchanged and confines adversarial optimization to the downstream decoder.

4.2 Generation-Aware Paired Supervision

Equation (2) aligns the distribution of decoded samples with the data distribution but does not constrain their sample-wise correspondence to the input latents. Optimized in isolation, the decoder

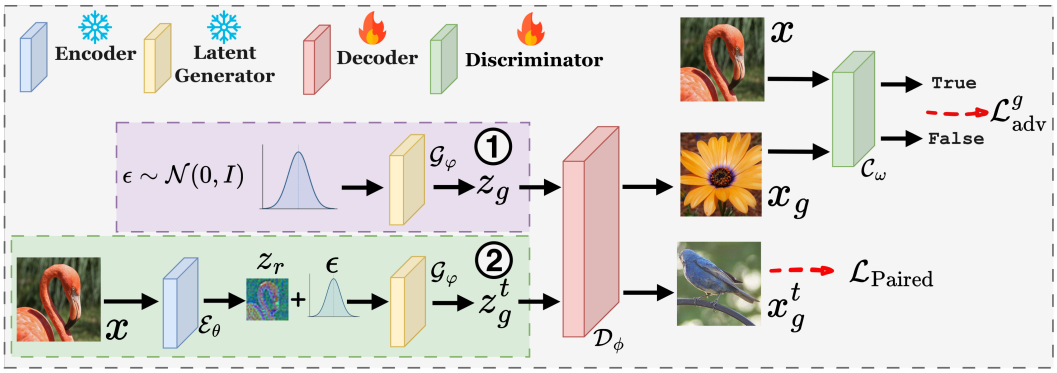

Figure 2: **Overview of Latent Prior GANs (LP-GANs).** LP-GANs freeze the pretrained latent generator and refine the decoder using two complementary objectives. **(1)** Standard generation produces $z_g \sim p_g$, and the decoded samples x_g are adversarially matched to real images through $\mathcal{L}^g_{\text{adv}}$. **(2)** Starting from a reconstruction latent z_r, we perturb and partially sample it with the frozen generator to obtain a generation-aware intermediate latent z^t_g that retains correspondence with x, enabling paired supervision through $\mathcal{L}_{\text{pair}}$. Together, these objectives adapt the decoder to generation-time latents while anchoring its refinement to the inherited latent-to-image mapping.

may therefore drift from its inherited latent-to-image mapping. Paired reconstruction and perceptual objectives can anchor this adaptation, but this requires source-image correspondence, which fully generated latents do not provide. We therefore construct generation-aware intermediate latents that move toward the generation-time distribution while remaining paired with their source images.

Given $x \sim p_{\text{data}}$, we first obtain $z_r = \mathcal{E}_\theta(x)$ and perturb it toward the sampler initialization as $z^t_{\text{noisy}} = a_t z_r + b_t \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$ and the noise level $\eta_t = b_t/(a_t + b_t)$ interpolates between z_r at $\eta_t = 0$ and the standard noise initialization at $\eta_t = 1$. Starting from z^t_{noisy}, we run the corresponding partial sampling procedure to obtain

$$z^t_g = \mathcal{S}_t\big(\mathcal{G}_\varphi, z^t_{\text{noisy}}, \xi\big) \sim p^t_g(z),$$

where $\mathcal{S}_t$ denotes the deployed sampler run from the corresponding noise level to the generation endpoint. As η_t increases, p^t_g moves toward the generation-time distribution. In particular, at $\eta_t = 1$, $z^t_{\text{noisy}} = \epsilon$ and the procedure reduces to standard generation, so $p^1_g = p_g$ (Fang et al., 2026). Because z^t_g originates from the reconstruction latent of x, it retains image-level correspondence with x.

We therefore define the paired objective as

$$\mathcal{L}_{\text{pair}}(\phi) = \mathbb{E}\Big[\mathcal{L}_{\text{rec}}\big(x, \mathcal{D}_\phi(z^t_g)\big) + \lambda_{\text{perc}}\, \mathcal{L}_{\text{perc}}\big(x, \mathcal{D}_\phi(z^t_g)\big)\Big], \tag{3}$$

where the expectation is over $x \sim p_{\text{data}}$, t, ϵ, and ξ, with t sampled uniformly during adaptation. The full decoder objective is

$$\mathcal{L}_{\text{LP-GANs}} = \mathcal{L}^g_{\text{adv}} + \lambda_{\text{pair}}\, \mathcal{L}_{\text{pair}}. \tag{4}$$

The adversarial term refines the decoder directly on the deployment distribution, while the paired term anchors this refinement to the inherited latent-to-image mapping.

5 Experiments

Our experiments cover two latent generation paradigms. We begin at the system level, where LP-GANs is compared with diffusion, flow, GAN, and autoregressive models (Section 5.1). The component analysis in Section 5.2 is on diffusion and flow generators: it separates the adversarial term from paired supervision, sweeps the trajectory noise level, and compares the four possible update scopes. Cost is the subject of Section 5.3, and Section 5.4 tests generators beyond the iMF family.

Experiment Setup. We conduct experiments on ImageNet-1K at 256 × 256 resolution (Deng et al., 2009). Our primary evaluation uses the iMF family (Geng et al., 2026), including iMF-B/2, iMF-M/2, iMF-L/2, and iMF-XL/2, all sharing the same pretrained SD-VAE tokenizer (Rombach et al., 2022). During LP-GANs training, the tokenizer encoder and latent generator remain frozen; the decoder is adapted jointly with a newly introduced discriminator, while all other components

Table 1: **System-level comparison on class-conditional ImageNet 256×256**. NFE denotes the number of function evaluations during inference. All numbers are with CFG when applicable. **Bold** indicates the best result in each comparison. † denotes results from (Geng et al., 2026), ‡ denotes results cited from GAR (Fang et al., 2026), and ⋆ denotes results reproduced or obtained by us under the official iMF protocol.

Method	# Params	NFE	FID
1-NFE diffusion/flow from scratch			
MeanFlow-XL/2⋆	676M	1	3.34
MeanFlow-XL/2+LP-GANs (Ours)⋆	676M	1	2.94
TiM-XL/2†	664M	1	3.26
α-Flow-XL/2+†	676M	1	2.58
iMF-B/2⋆	89M	1	3.37
iMF-M/2⋆	174M	1	2.27
iMF-L/2⋆	409M	1	1.86
iMF-XL/2⋆	610M	1	1.73
iMF-B/2 + GAR‡	89M	1	2.81
iMF-M/2+ GAR‡	174M	1	1.98
iMF-L/2+ GAR‡	409M	1	1.58
iMF-XL/2 + GAR‡	610M	1	1.50
iMF-B/2 + LP-GANs (Ours)⋆	89M	1	**2.08**
iMF-M/2 + LP-GANs (Ours)⋆	174M	1	**1.73**
iMF-L/2 + LP-GANs (Ours)⋆	409M	1	**1.48**
iMF-XL/2 + LP-GANs (Ours)⋆	610M	1	**1.32**
2-NFE diffusion/flow from scratch			
IMM-XL/2†	675M	1×2	7.77
MeanFlow-XL/2+†	676M	2	2.20
α-Flow-XL/2+†	676M	2	1.95
iMF-XL/2†	610M	2	1.54
1-NFE diffusion/flow (distillation)			
π-Flow-XL/2†	675M	1	2.85
DMF-XL/2+⋆	675M	1	2.22
DMF-XL/2+ + LP-GANs (Ours)⋆	675M	1	1.91
FACM-XL/2†	675M	1	1.76
Multi-NFE diffusion/flow			
ADM-G†	554M	250×2	4.59
LDM-4-G†	400M	250×2	3.60
SimDiff†	2B	1000×2	2.77
DiT-XL/2†	675M	250×2	2.27
SiT-XL/2†	675M	250×2	2.06
SiT-XL/2 + REPA†	675M	250×2	1.42
SiD2†	–	512×2	1.38
LightningDiT-XL/2†	675M	250×2	1.35
DDT-XL/2†	675M	250×2	1.26
RAE + DiT^{DH}-XL†	839M	250×2	1.13
GANs			
BigGAN†	112M	1	6.95
GigaGAN†	569M	1	3.45
StyleGAN-XL†	166M	1	2.30
autoregressive/masking			
JetFormer-L†	2.75B	256×2	6.64
MaskGIT†	227M	8	6.18
TiTok-L32⋆	307M	8×2	2.65
TiTok-L32 + LP-GANs (Ours)⋆	307M	8×2	1.90
RQ-Transformer†	3.8B	256×2	3.80
STARFlow†	1.4B	1024×2	2.40
LLamaGen-3B†	3.1B	256×2	2.18
VAR-$d30$†	2B	10×2	1.92
MAR-H†	943M	256×2	1.55
RAR-XXL†	1.5B	256×2	1.48
xAR-H†	1.1B	50×2	1.24

stay frozen. Generation-time latents are sampled from the frozen latent generator using classifier-free guidance (CFG), so that decoder refinement operates under the CFG-induced latent distribution. CFG is likewise used to construct generation-aware intermediate latents for paired supervision. Beyond the iMF family, we further apply LP-GANs to MeanFlow-XL/2 (Geng et al., 2025) and DMF-XL/2+ (Lee et al., 2026). We additionally evaluate LP-GANs on TiTok-L32 (Yu et al., 2024b) to test whether the decoder-refinement principle extends to autoregressive latent generation. Additional implementation details are provided in Appendix A.

Evaluation Metrics and Protocols. We evaluate generative performance using Fréchet Inception Distance (FID) (Heusel et al., 2017), referred to as gFID for generated samples. Notably, iMF (Geng et al., 2026) and MeanFlow (Geng et al., 2025) adopt a distinct FID evaluation protocol, which we call the *official iMF protocol*, whereas the OpenAI protocol is more widely used. Unless a table states otherwise, gFID is computed under the OpenAI protocol, and we only compare numbers that share a protocol. The same model gets different absolute gFID values under the two protocols; Appendix C explains the protocol differences.

5.1 System-Level Comparison

System-Level Comparison Baselines. We present a system-level comparison on class-conditional ImageNet 256×256 across diffusion/flow, GAN, and autoregressive or masked generative models, with inference cost measured by the number of function evaluations (NFE). For diffusion and flow models trained from scratch, we include 1-NFE methods MeanFlow-XL/2 (Geng et al., 2025), TiM-XL/2 (Wang et al., 2025), α-Flow-XL/2+ (Zhang et al., 2025), and iMF (Geng et al., 2026) at multiple scales (B/2, M/2, L/2, XL/2), together with 2-NFE methods IMM-XL/2 (Zhou et al., 2025a), MeanFlow-XL/2+ (Geng et al., 2025), α-Flow-XL/2+ (Zhang et al., 2025), and iMF-XL/2. We further include distillation-based 1-NFE methods π-Flow-XL/2 (Chen et al., 2025a), DMF-XL/2+ (Lee et al., 2026), and FACM-XL/2 (Peng et al., 2026), as well as multi-NFE diffusion/flow models ADM-G (Dhariwal & Nichol, 2021), LDM-4-G (Rombach et al., 2022), SimDiff (Hoogeboom et al., 2023), DiT-XL/2 (Peebles & Xie, 2023), SiT-XL/2 (Ma et al., 2024), SiT-XL/2+REPA (Yu et al., 2025), SiD2 (Hoogeboom et al., 2025), LightningDiT-XL/2 (Yao & Wang, 2025), DDT-XL/2 (Wang et al., 2026), and RAE+DiT^{DH}-XL (Zheng et al., 2026). For broader reference, we also compare with GAN-based methods BigGAN (Brock et al., 2018), GigaGAN (Kang et al., 2023), and StyleGAN-XL (Sauer et al., 2022), and autoregressive or masked models JetFormer-L (Tschannen et al., 2025), MaskGIT (Chang et al., 2022), TiTok-L32 (Yu et al., 2024b), RQ-Transformer (Lee et al., 2022), STARFlow (Gu et al., 2025), LLamaGen-3B (Sun et al., 2024), VAR-$d30$ (Tian et al., 2024), MAR-H (Li et al., 2024), RAR-XXL (Yu et al., 2024a), and xAR-H (Ren et al., 2025).

System-Level Generative Performance. We first compare LP-GANs with other class-conditional image generators at the system level. As shown in Table 1, LP-GANs consistently improve generative performance across all four evaluated latent generators without changing their inference cost. For the iMF family, LP-GANs reduce gFID from 3.37 to 2.08 on B/2, 2.27 to 1.73 on M/2, 1.86 to 1.48 on L/2, and 1.73 to 1.32 on XL/2, while retaining one-step generation. The same improve-

Table 2: **LP-GANs across iMF model scales.** We report gFID without and with CFG across iMF model scales, comparing the pretrained baseline, LP-GANs without paired supervision (PS), and full LP-GANs under both evaluation protocols; full LP-GANs consistently perform best. ‡: our reproduction using the official iMF protocol. ⋆: our reproduction using the OpenAI protocol. See Appendix C for protocol details.

Setting	gFID w/o CFG (↓)				gFID w/ CFG (↓)			
	iMF-B/2	iMF-M/2	iMF-L/2	iMF-XL/2	iMF-B/2	iMF-M/2	iMF-L/2	iMF-XL/2
Baseline‡	16.58	11.92	9.28	9.78	3.37	2.27	1.86	1.73
LP-GANs w/o PS‡	19.43	15.58	11.92	11.96	2.55	2.07	1.78	1.69
LP-GANs w/ PS‡	**13.39**	**11.20**	**8.49**	**8.51**	**2.08**	**1.73**	**1.48**	**1.32**
Baseline⋆	16.41	11.92	9.26	9.41	3.47	2.40	1.90	1.80
LP-GANs w/o PS⋆	18.99	15.11	11.63	11.12	2.56	2.06	1.82	1.67
LP-GANs w/ PS⋆	**13.64**	**10.66**	**8.52**	**8.52**	**2.14**	**1.77**	**1.51**	**1.51**

ment extends beyond iMF: MeanFlow-XL/2 improves from 3.34 to 2.94, while the distillation-based DMF-XL/2+ improves from 2.22 to 1.91, both at NFE $= 1$. LP-GANs also transfer to autoregressive latent generation, reducing TiTok-L32 gFID from 2.65 to 1.90 while preserving its NFE $= 8$ inference procedure. Among the evaluated 1-NFE diffusion/flow models, iMF-XL/2 + LP-GANs achieves the lowest gFID of 1.32. Together, these results show that decoder refinement provides consistent gains across model scales, generator architectures, and both diffusion/flow and autoregressive latent generation, without additional inference NFEs.

Qualitative Results. Finally, we qualitatively examine the effect of decoder refinement on generated samples; selected matched samples are provided in Appendix E (Figures 5–14). In the baseline comparisons, samples in the same column share identical generated latents and the two rows differ only in the decoder. Compared with the original pretrained decoder, LP-GANs refinement produces visually cleaner and more detailed samples, with the main subject and broad layout appearing similar in many of the examples shown. The matched latents let us compare the decoders directly, but they do not rule out changes to the image content.

5.2 Component Analysis on Diffusion- and Flow-Based Generators

We study LP-GANs on diffusion- and flow-based latent generative models, where the latent generator is frozen and decoder refinement is performed on CFG-induced generation-time latents. We first isolate the effects of generation-time adversarial refinement and generation-aware paired supervision across four iMF model scales ranging from B/2 to XL/2, then analyze sensitivity to the trajectory noise level controlling source correspondence, and finally compare decoder-only, generator-only, and joint update scopes for the inherited latent prior.

Effect of Adversarial Refinement and Paired Supervision. Table 2 isolates the effects of generation-time adversarial refinement and generation-aware paired supervision (PS) across four iMF model scales. With CFG evaluation, adversarial refinement alone (LP-GANs w/o PS) consistently improves over the pretrained baseline under both evaluation protocols, showing that directly optimizing the decoder on fully generated latents is effective even without paired supervision. Adding generation-aware paired supervision further improves gFID at every scale: under the official iMF protocol, gFID decreases from 2.55 to 2.08 on B/2, 2.07 to 1.73 on M/2, 1.78 to 1.48 on L/2, and 1.69 to 1.32 on XL/2. The behavior differs without CFG: adversarial refinement alone degrades gFID, indicating that optimizing the decoder solely on fully generated latents lets it drift from the inherited latent-to-image mapping; adding paired supervision prevents this drift and recovers and improves upon the pretrained baseline across all four scales. This contrast directly confirms the anchoring role of the paired objective in Equation. 4. The OpenAI and official iMF protocols yield close but non-identical gFID values, so all comparisons are made within the same evaluation protocol.

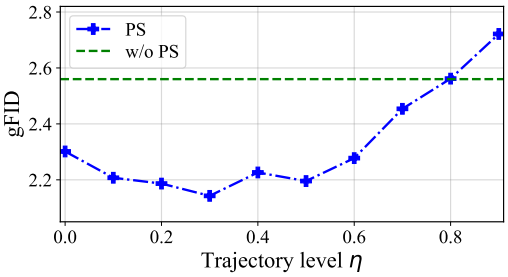

Figure 3: CFG gFID across trajectory level η. The dashed line is LP-GANs w/o PS (2.56); the solid curve is best at $\eta = 0.3$.

Sensitivity to the Trajectory Noise Level. Generation-aware paired supervision depends on the trajectory noise level η_t, which controls the trade-off between proximity to generation-time latents and correspondence with the source image. We therefore perform a sensitivity analysis on iMF-B/2. As depicted in Figure 3, for $\eta_t \in [0, 0.7]$, adding paired supervision consistently improves over adversarial refinement alone, with the best performance obtained at $\eta_t = 0.3$. Performance deteriorates when $\eta_t \geq 0.8$. This behavior is consistent with the reconstruction-to-generation trajectory characterized in GAR (Fang et al., 2026): as η_t increases, intermediate latents become increasingly generation-like while their correspondence with the source image progressively weakens. At very high noise levels, this correspondence becomes too weak for reconstruction and perceptual objectives to provide reliable sample-wise supervision, causing these paired constraints to become detrimental rather than beneficial.

Which Component Should Be Refined? We compare four update scopes for the inherited latent prior: frozen, decoder-only, generator-only, and joint refinement. As shown in Table 3, under CFG decoder-only refinement performs best, followed by joint refinement and the frozen baseline, and generator-only performs worst; without CFG, decoder-only still comes out first and generator-only last. This fits our diagnosis in Sections 3 and 4 that the mismatch sits at the decoder, not in the latent distribution. p_g is already the distribution the model sees at deployment, so changing the generator is unnecessary; in our runs it is also harmful, because the updated p_g moves away from the distribution the decoder was trained to decode, and generator-only in fact ranks below even the frozen baseline. Adapting the decoder to p_g with the generator kept frozen is already sufficient. The three trainable scopes have comparable costs (12.7, 11.5, and 13.9 H100 GPU-hours per epoch for decoder-only, generator-only, and joint refinement): generator-only is marginally the cheapest but performs worst. Decoder-only attains the best gFID at near-minimal cost, and joint refinement is both more expensive and inferior. We therefore use decoder-only refinement throughout LP-GANs.

Table 3: **Which component to refine?** On iMF-B/2 under the OpenAI protocol, decoder-only refinement achieves the best gFID at near-minimal cost. CFG/no-CFG are gFID; cost is H100-h/epoch. Lower is better.

Scope	CFG	no-CFG	Cost
Frozen	3.47	16.41	0
Decoder-only	**2.14**	**13.64**	12.7
Generator-only	3.91	22.33	**11.5**
Joint	2.40	17.25	13.9

5.3 Training Efficiency

We quantify the cost of decoder-only LP-GANs refinement in two ways: against training class-conditional GANs from scratch and against pretraining the tokenizer and latent generator.

Efficiency Compared with From-Scratch GAN Training. To contextualize the adversarial training burden of LP-GANs, we compare their training cost and generative performance with representative GAN systems, including the BigGAN and StyleGAN families trained on ImageNet, as shown in Table 4. Conventional GANs rely on adversarial optimization to learn and refine generative structure from scratch, requiring substantially more adversarial training. LP-GANs instead inherit this structure from a pretrained latent generative model and restrict adversarial optimization to decoder refinement. As a result, LP-GANs converge within only five epochs of decoder refinement and achieve stronger generative performance at a total cost of 88 H100 GPU-hours for iMF-XL/2, versus an estimated 2,000 to 5,000 GPU-hours for the from-scratch GAN baselines. This comparison supports the central motivation of LP-GANs: once generative structure has already been learned, adversarial optimization can be repurposed from structure discovery to lightweight refinement.

Table 4: **From-scratch GANs vs. LP-GANs.** Class-conditional ImageNet 256, one-step. Costs are estimated total wall-clock GPU-hours on a single H100 GPU. LP-GANs cost covers decoder-only refinement only and excludes latent-prior pretraining.

Method	FID ↓	H100-h ↓
BigGAN(Brock et al., 2018)	6.95	5,000
StyleGAN-XL(Sauer et al., 2022)	2.30	2,000
LP-GANs (iMF-XL/2)	**1.32**	**88**

Refinement Cost Relative to Latent-Prior Training. We further place the cost of LP-GANs refinement in the context of constructing the latent prior itself. As shown in Table 5, we compare the training cost of the pretrained SD-VAE tokenizer, the iMF latent generators at B/2, M/2, L/2, and XL/2 scales, and the subsequent LP-GANs refinement. Across model scales, LP-GANs require only a small fraction of the computation used to train the tokenizer and latent generator. Thus, the

improvement does not require relearning the latent prior, but instead adds a lightweight refinement stage on top of an already trained generative system. This further demonstrates the computational advantage of inheriting rather than rediscovering generative structure.

5.4 Generalization to Other Latent Generators

Generalization Across Latent Generators. We further evaluate LP-GANs on MeanFlow-XL/2 (Geng et al., 2025) and DMF-XL/2+ (Lee et al., 2026). Across both the OpenAI and official iMF evaluation protocols, LP-GANs consistently improve gFID over the pretrained baselines (Table 6): MeanFlow-XL/2 improves from 3.49 to 2.98 and from 3.34 to 2.94, and DMF-XL/2+ from 2.40 to 1.95 and from 2.22 to 1.91, under the OpenAI and official protocols, respectively. These results show that the benefits of generation-time decoder refinement extend beyond the iMF family and generalize across different diffusion- and flow-based latent generators.

Table 5: **Latent-prior pretraining vs. LP-GANs refinement.** Costs are estimated total wall-clock GPU-hours on a single H100 GPU; tokenizer and latent-generator costs are FLOPs-based estimates.

Model	Tokenizer	Latent Generator	LP-GANs
iMF-B/2	750	440	**64**
iMF-M/2	750	860	**66**
iMF-L/2	750	2,000	**72**
iMF-XL/2	750	3,800	**88**

Generalization to Autoregressive Latent Generators. We further evaluate whether LP-GANs generalize beyond diffusion- and flow-based latent generation to autoregressive latent generators. Unlike diffusion and flow models, autoregressive generators do not naturally provide the reconstruction-to-generation trajectory used above to construct generation-aware intermediate latents. We therefore retain generation-time adversarial refinement on fully generated latents, while using encoder-induced reconstruction latents $z_r = \mathcal{E}_\theta(x)$ for paired reconstruction and perceptual supervision. Concretely, we adapt only the decoder of TiTok-L32 (Yu et al., 2024b), keeping its eight-step MaskGIT sampling unchanged. This preserves the core LP-GANs principle: adversarially refining the decoder on the latent distribution encountered at generation time while anchoring it to the inherited latent-to-image mapping through paired supervision.

Table 6: **Generalization to MeanFlow-XL/2 and DMF-XL/2+.** CFG gFID under both evaluation protocols; lower is better.

Generator	Protocol	Baseline	LP-GANs
MeanFlow-XL/2	OpenAI	3.49	**2.98**
	Official	3.34	**2.94**
DMF-XL/2+	OpenAI	2.40	**1.95**
	Official	2.22	**1.91**

As shown in Figure 4, LP-GANs consistently improve TiTok-L32 under both evaluation protocols. Under the OpenAI protocol, gFID decreases from 2.81 to 1.98, while under the official iMF protocol it decreases from 2.65 to 1.90. Since the latent generator and its inference procedure remain unchanged, these gains arise solely from decoder refinement. Importantly, this result shows that LP-GANs are not tied to diffusion- or flow-based latent generators: the underlying decoder-refinement principle also transfers to autoregressive latent generation.

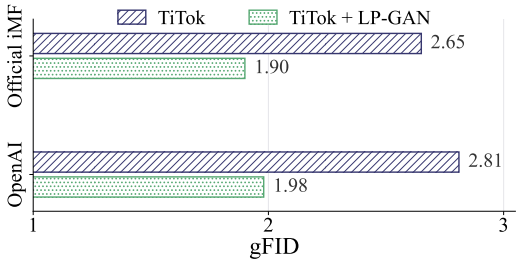

Figure 4: TiTok-L32 gFID under both protocols.

6 Conclusion

We introduced **Latent Prior GANs (LP-GANs)**, a framework for efficient decoder refinement via inherited generative structure. With a strong latent generative model available, adversarial learning no longer has to build generative structure from scratch. It only improves generation under the inherited latent prior. The decoder is then the part that needs to change: it was trained on reconstruction latents, but at generation time it receives generated latents. LP-GANs directly address this decoder train-generation mismatch by adapting the decoder to generation-time latents, anchored by generation-aware paired supervision. We observe the same result for every generator and scale we test: five epochs of adversarial adaptation already give better samples than the pretrained model. To go beyond ImageNet, we would need stronger pretrained models. We hope this recipe then applies to more generators and other datasets.

7 AI USE STATEMENT

Generative AI tools were used in this work solely to assist with language refinement and editorial improvements. They did not contribute to the development of the research ideas, methodological design, theoretical derivations, experimental setup, data preparation, implementation, or interpretation of the scientific results. Any AI-assisted wording was carefully checked and edited by the authors. The authors remain fully responsible for the accuracy, integrity, and final presentation of the manuscript.

8 ETHICS STATEMENT

This research does not involve human-subject studies, user studies, crowdsourcing, participant interaction, or the collection of new personal or sensitive data. Therefore, Institutional Review Board (IRB) approval or an equivalent ethics review was not required.

Our experiments are conducted using existing publicly available datasets and model architectures, including ImageNet. The manuscript presents a limited number of images generated by generative models trained on ImageNet for qualitative visualization and analysis of the experimental results. We do not collect additional information about the depicted content, attempt to identify individuals, or conduct analyses related to personal attributes.

This work does not introduce a new dataset or collect new human-subject data. We are not aware of additional ethical risks introduced by the proposed methodology beyond those associated with the publicly available datasets and models used in our experiments. All resources are used for research purposes in accordance with their applicable terms and conditions.

9 REPRODUCIBILITY STATEMENT

We have taken several steps to support the reproducibility and independent verification of the empirical results reported in this work. The supplementary material includes the implementation code used in our experiments, together with detailed instructions for running the code and reproducing the main results.

We also document the experimental setup, including the datasets and preprocessing procedures, model configurations, training settings, optimization hyperparameters, implementation details, computational resources, and evaluation protocols, in Appendix A. These materials are intended to provide sufficient information for independently reproducing the reported experiments and verifying the main empirical findings.

Upon publication, we will publicly release the complete codebase and the model checkpoints used in this work, together with the corresponding configuration files and execution instructions. We expect these resources to further facilitate reproducibility and future research building on our results.

REFERENCES

Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *ICLR*, 2018.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 9630–9640. IEEE, 2021.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. In *CVPR*, 2022.

Hansheng Chen, Kai Zhang, Hao Tan, Leonidas J. Guibas, Gordon Wetzstein, and Sai Bi. pi-flow: Policy-based few-step generation via imitation distillation. *ArXiv*, 2025a.

Hao Chen, Ze Wang, Xiang Li, Ximeng Sun, Fangyi Chen, Jiang Liu, Jindong Wang, Bhiksha Raj, Zicheng Liu, and Emad Barsoum. Softvq-vae: Efficient 1-dimensional continuous tokenizer. In *CVPR*, 2025b.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, K. Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

Xianghong Fang, Wenjie Shu, Tongda Xu, Wenlong Mou, Dehan Kong, and Tim G. J. Rudner. Bridging reconstruction and generation: A latent distribution perspective on evaluation and improvement. *arXiv preprint arXiv:2609.24088*, 2026.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *NeurIPS*, 2025.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J. Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models. In *CVPR*, 2026.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

Jiatao Gu, Tianrong Chen, David Berthelot, Huangjie Zheng, Yuyang Wang, Ruixiang Zhang, Laurent Dinh, Miguel Angel Bautista, Joshua M. Susskind, and Shuangfei Zhai. Starflow: Scaling latent normalizing flows for high-resolution image synthesis. In *NeurIPS*, 2025.

Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *NeurIPS*, 2017.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *ICML*, 2023.

Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler diffusion (sid2): 1.5 fid on imagenet512 with pixel-space diffusion. In *CVPR*, 2025.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *CVPR*, 2023.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *NeurIPS*, 2021.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *CVPR*, 2022.

Kyungmin Lee, Sihyun Yu, and Jinwoo Shin. Decoupled meanflow: Turning flow models into flow maps for accelerated sampling. In *ICLR*, 2026.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. In *NeurIPS*, 2024.

Xiang Li, Kai Qiu, Hao Chen, Jason Kuen, Jiuxiang Gu, Bhiksha Raj, and Zhe Lin. Imagefolder: Autoregressive image generation with folded tokens. In *ICLR*, 2025.

Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017.

Shanchuan Lin, Anran Wang, and Xiao Yang. Sdxl-lightning: Progressive adversarial diffusion distillation. In *ECCV*, 2024.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11966–11976. IEEE, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas Matthew Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *ECCV*, 2024.

Lars M. Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *ICML*, 2018.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

William S. Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

Yansong Peng, Kai Zhu, Yu Liu, Ping Wu, Hebei Li, Xiaoyan Sun, and Feng Wu. Flow-anchored consistency models. In *ICLR*, 2026.

Kai Qiu, Xiang Li, Hao Chen, Jason Kuen, Xiaohao Xu, Jiuxiang Gu, Yinyi Luo, Bhiksha Raj, Zhe Lin, and Marios Savvides. Image tokenizer needs post-training. *ArXiv*, 2025.

Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *NeurIPS*, 2019.

Sucheng Ren, Qihang Yu, Ju He, Xiaohui Shen, Alan Yuille, and Liang-Chieh Chen. Beyond next-token: Next-x prediction for autoregressive visual generation. In *ICCV*, 2025.

Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *SIGGRAPH*, 2022.

Axel Sauer, Dominik Lorenz, A. Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *ECCV*, 2023.

Axel Sauer, Frederic Boesel, Tim Dockhorn, A. Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation. In *SIGGRAPH Asia*, 2024.

Dazhong Shen, Guanglu Song, Yi Zhang, Bingqi Ma, Lujundong Li, Dongzhi Jiang, Zhuofan Zong, and Yu Liu. Adt: Tuning diffusion models with adversarial supervision. *ArXiv*, 2025.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *ArXiv*, 2024.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *NeurIPS*, 2024.

Michael Tschannen, André Susano Pinto, and Alexander Kolesnikov. Jetformer: An autoregressive generative model of raw images and text. In *ICLR*, 2025.

Hung-Yu Tseng, Lu Jiang, Ce Liu, Ming-Hsuan Yang, and Weilong Yang. Regularizing generative adversarial networks under limited data. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7917–7927. IEEE, 2021.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017.

Shuai Wang, Zhichao Tian, Weilin Huang, and Limin Wang. Ddt: Decoupled diffusion transformer. In *CVPR*, 2026.

Zidong Wang, Yiyuan Zhang, Xiaoyu Yue, Xiangyu Yue, Yangguang Li, Wanli Ouyang, and Lei Bai. Transition models: Rethinking the generative learning objective. *ArXiv*, 2025.

Yanwu Xu, Yang Zhao, Zhisheng Xiao, and Tingbo Hou. Ufogen: You forward once large scale text-to-image generation via diffusion gans. In *CVPR*, 2024.

Jingfeng Yao and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *CVPR*, 2025.

Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Randomized autoregressive visual generation. In *ICCV*, 2024a.

Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. In *NeurIPS*, 2024b.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *ICLR*, 2025.

Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency regularization for generative adversarial networks. *arXiv preprint arXiv:1910.12027*, 2019.

Huijie Zhang, Aliaksandr Siarohin, Willi Menapace, Michael Vasilkovsky, Sergey Tulyakov, Qing Qu, and Ivan Skorokhodov. Alphaflow: Understanding and improving meanflow models. *ArXiv*, 2025.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *Advances in neural information processing systems*, 33:7559–7570, 2020.

Bowen Zheng and Tianming Yang. Revisiting diffusion models: From generative pre-training to one-step generation. In *ICML*, 2025.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *ICLR*, 2026.

Linqi Zhou, Stefano Ermon, and Jiaming Song. Inductive moment matching. In *ICML*, 2025a.

Mingyuan Zhou, Huangjie Zheng, Yi Gu, Zhendong Wang, and Hai Huang. Adversarial score identity distillation: Rapidly surpassing the teacher in one step. In *ICLR*, 2025b.

A Implementation and Training Details

Data and preprocessing. All experiments use ImageNet-1K (Russakovsky et al., 2015). To reach 256×256, images are resized with the iterative box-downsampling procedure used in LlamaGen (Sun et al., 2024), followed by random cropping and horizontal flipping during training.

Training setup. The VAE encoder and latent generator remain frozen throughout training, while the decoder and a DINO-based image discriminator are optimized. Both the generated-latent and paired branches produce latents with classifier-free guidance during training. The iMF, MeanFlow, and DMF runs train for five epochs on eight NVIDIA H100 GPUs with batch size 8 per GPU (4 for the η-sweep and update-scope ablation on iMF-B/2), using single-step (NFE$=1$) generation. The decoder and discriminator are optimized separately with AdamW (Loshchilov & Hutter, 2017), $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight decay 10^{-4}. Both learning rates start at 10^{-5}; the decoder rate is halved after each epoch down to 10^{-6}, while the discriminator rate stays at 10^{-5}. Each batch runs one decoder update and then one discriminator update, with the discriminator active from the first epoch; gradients are clipped at norm 1.0 for both optimizers.

Losses. We use hinge losses (Lim & Ye, 2017) for the discriminator and decoder adversarial updates. The paired branch combines pixel-space MSE with a perceptual loss. The perceptual loss averages VGG-based LPIPS (Zhang et al., 2018) and ImageNet-pretrained ConvNeXt-S features (Liu et al., 2022) with weights 1.0 and 0.1; the ConvNeXt-S term is an MSE on the final classifier features after bicubic resizing to 224×224. We set the adversarial coefficient to 0.7 times the ratio of paired to adversarial gradient norms at the decoder's last layer.

Discriminator. The discriminator is a ViT-S/16 pretrained with DINO (Caron et al., 2021) following the VAR setup (Tian et al., 2024), with LeCAM (Tseng et al., 2021) (0.001), consistency regularization (Zhang et al., 2019) (4.0), and differentiable DiffAugment (Zhao et al., 2020) (color, translation, and cutout at strength 0.5).

Noise levels and external generators. MeanFlow-XL/2 (Geng et al., 2025) and DMF-XL/2+ (Lee et al., 2026) use the same decoder-adaptation recipe as the iMF models, with the latent generator frozen. We use the fixed noise level $\eta = 0.3$ for the reported iMF-B/2 model and the update-scope ablation, $\eta = 0.2$ for iMF-M/2, iMF-L/2, iMF-XL/2, and MeanFlow-XL/2, and $\eta = 0.4$ for DMF-XL/2+. The classifier-free guidance settings for one-step generation are in Table 7.

TiTok-L32 configuration. We use the pretrained TiTok-L32 model with the `titok-vql32` tokenizer, 32 latent tokens, and native resolution 256×256. Its MaskGIT (Chang et al., 2022) generator stays frozen; it uses eight sampling steps and CFG scale 4.5 to produce the generated tokens for the adversarial branch. Training runs on eight NVIDIA H100 GPUs with batch size 2 per GPU for five epochs. We use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, zero weight decay, and learning rates 10^{-5} halved after each epoch down to 10^{-6}; the paired and adversarial loss weights are both set to 1. Each batch performs one decoder update followed by one discriminator update. TiTok-L32 uses no trajectory-level η: real tokenizer tokens are paired with source images, and generated autoregressive tokens form the fake-distribution branch.

Table 7: **Classifier-free guidance settings.** Guidance scale ω and interval $[t_{\min}, t_{\max}]$ for one-step generation; $\omega = 1$ means no sampling-time guidance (MeanFlow-XL/2 bakes guidance into training, DMF-XL/2+ is released without it).

Generator	ω	$t_{\min}$	$t_{\max}$
iMF-B/2	8.0	0.40	0.65
iMF-M/2	10.5	0.40	0.60
iMF-L/2	8.0	0.44	0.60
iMF-XL/2	5.25	0.48	0.65
MeanFlow-XL/2	1.0	0.00	1.00
DMF-XL/2+	1.0	0.00	1.00

Checkpoint selection and evaluation. We report each adaptation value from a saved epoch checkpoint. We save a checkpoint at the end of every epoch. The value reported in a table comes from the checkpoint with the lowest gFID under that table's protocol and sampling setting, evaluated both with and without CFG. A checkpoint is never chosen across models or protocols. For evaluation, we generate 50,000 images with a fixed class-conditional seed: one-step generation for the diffusion and flow models, and the eight-step procedure for TiTok-L32.

Compute accounting. Training cost excludes evaluation, checkpointing, and latent encoding. On iMF-XL/2, decoder-only refinement takes an estimated 17.6 H100 GPU-hours per epoch, or 88 H100 GPU-hours in total over five epochs. The per-epoch update-scope costs measured on iMF-B/2 (Table 3) are 12.7, 11.5, and 13.9 H100 GPU-hours for decoder-only, generator-only, and joint refinement, respectively; the three variants share the same training recipe and differ only in the set of

trainable parameters. We note that the tokenizer and generator costs are not our own measurements: they come from FLOPs counts or the source papers. The BigGAN and StyleGAN-XL numbers were produced on Google TPUs and V100 GPUs, and as such they do not sit on the same footing as our H100 timings.

B LIMITATIONS

Our results are confined to class-conditional ImageNet at 256×256, and we have not tested text-to-image generation, higher resolutions, or datasets beyond ImageNet. The noise level η is another setting we cannot fix in advance: values between 0.2 and 0.4 behave similarly in our runs, but the best point moves with the model and the sampler (Table 9), and we currently choose it by sweeping. The approach also relies on a pretrained latent model strong enough that the generative structure is already in place; where such a model is missing, there is little to inherit and the method reduces to ordinary GAN training.

C GFID EVALUATION PROTOCOL

Table 8: Evaluation protocols on the iMF-B/2 baseline. The official iMF protocol uses the torch-fidelity pipeline and `jit_in256_stats.npz`; the OpenAI protocol uses TensorFlow Inception and `VIRTUAL_imagenet256_labeled.npz`.

Protocol	Feature extractor	Reference statistics	gFID
Official iMF	torch-fidelity	`jit_in256_stats.npz`	3.39
Official iMF (reproduced)	torch-fidelity	`jit_in256_stats.npz`	3.37
OpenAI	TensorFlow Inception	`VIRTUAL_imagenet256_labeled.npz`	3.47

The official iMF-B/2 evaluation (Geng et al., 2026) differs from the commonly used OpenAI protocol in three aspects: (i) the computational platform (TPU vs. GPU), (ii) the Inception feature implementation and weights (the PyTorch torch-fidelity pipeline vs. TensorFlow Inception), and (iii) the reference real-image statistics (`jit_in256_stats.npz` vs. `VIRTUAL_imagenet256_labeled.npz`). Absolute gFID values are not directly comparable across protocols, so all comparisons are made within the same protocol and reference statistics.

D ADDITIONAL EXPERIMENTAL RESULTS

D.1 TRAJECTORY-LEVEL SENSITIVITY

Figure 3 and Table 9 give the full CFG gFID sweep over η that we used to select the iMF-B/2 model. The row at 2.56 is adversarial refinement without paired supervision and serves as the reference. The paired term already helps at $\eta = 0$, where gFID is 2.30, and it keeps helping all the way through $\eta = 0.7$; we obtain the lowest value, 2.14, at $\eta = 0.3$. After that the gain goes away: gFID meets the reference again at $\eta = 0.8$, reaches 2.72 at 0.9, and ends at 8.67 at 1.0. We think this is expected: as η increases, the latents look more like generated latents and less like the source image, so at some point the paired losses no longer have a reliable image to compare against (Fang et al., 2026).

Table 9: **Trajectory-level sensitivity on iMF-B/2.** All values are CFG gFID (↓) under the OpenAI protocol.

Trajectory level η	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
CFG gFID ↓	2.30	2.21	2.19	**2.14**	2.23	2.20	2.28	2.45	2.56	2.72	8.67

D.2 GENERALIZATION GFID

Table 6 summarizes gFID under both protocols for the one-step generators MeanFlow-XL/2 and DMF-XL/2+, and Figure 4 for the autoregressive TiTok-L32. LP-GANs improve gFID in every setting, with relative reductions of 12 to 30%.

E QUALITATIVE COMPARISONS

We show selected matched samples for the main iMF-XL/2 comparison, the paired-supervision ablation, the smaller iMF-M/2 scale, and three other latent generators (MeanFlow-XL/2, DMF-XL/2+, TiTok-L32). Each figure has two rows of eight samples. The top row (*before*) shows images produced from the generative structure of the original latent prior; the bottom row (*after*) shows images from the same latent prior after LP-GANs refinement. Samples in the same column share latents and conditioning, so the two rows differ only in the decoder. All samples use the classifier-free guidance settings in Table 7. In the paired-supervision ablation below, before and after instead denote refinement without and with paired supervision. The figures are illustrative and not part of the quantitative metrics.

E.1 iMF-XL/2: BASELINE VERSUS LP-GANs

Figures 5–7 compare the pretrained iMF-XL/2 decoder with the LP-GANs adapted decoder on three classes. Baseline samples show local artifacts and over-smoothed or garbled textures; LP-GANs give sharper details and fewer artifacts while keeping scene layout and class identity.



Figure 5: Class 980: volcano (top: before, bottom: after).



Figure 6: Class 979: valley (top: before, bottom: after).



Figure 7: Class 88: macaw (top: before, bottom: after).

E.2 iMF-XL/2: Effect of Paired Supervision

Figures 8–10 isolate paired supervision, contrasting adversarial refinement alone (top) with the full LP-GANs objective (bottom). The top rows, with adversarial refinement alone, do clear up artifacts, yet a number of samples move off their conditioning and identity or structure changes. Once the paired term is added (bottom rows), the decoder is tied back to the inherited mapping, which keeps identity and structure in place; the detail gains from the adversarial term still come through.



Figure 8: Class 22: bald eagle (top: before, bottom: after).



Figure 9: Class 94: hummingbird (top: before, bottom: after).



Figure 10: Class 975: lakeside (top: before, bottom: after).

E.3 Other Latent Generators

Figures 11–14 show that the same gains hold beyond iMF-XL/2. They cover the smaller iMF-M/2 scale and three other generators: MeanFlow-XL/2, DMF-XL/2+, and TiTok-L32.



Figure 11: iMF-M/2: eight selected classes (top: before, bottom: after).

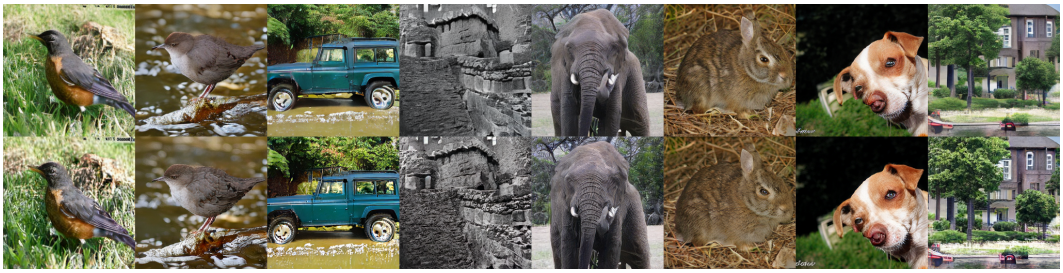

Figure 12: DMF-XL/2+: eight selected classes (top: before, bottom: after).



Figure 13: MeanFlow-XL/2: eight selected classes (top: before, bottom: after).



Figure 14: TiTok-L32: eight selected classes (top: before, bottom: after).