Magnitude-preserving Discriminators Enable Efficient GANs

Nick Yiwen Huang **Jackson Woodleigh*** **Aaron Gokaslan*†** **Xinjie Yi‡** **James Tompkin**
Brown University †Modal ‡Independent

github.com/brownvc/R3GAN2

Abstract

GANs are appealing because they generate sharp images with a single forward execution. Recent works have stabilized GAN training, but FID may still plateau on diverse datasets like ImageNet-256 even as training remains stable. We identify severe growth of discriminator activation magnitudes as a hidden pathology accompanying this plateau. Our analysis shows that adversarial training permits the discriminator to increase empirical real–fake separation by amplifying already-discriminative responses, without necessarily improving the underlying representation. Motivated by this observation, we redesign R3GAN's network components to preserve activation magnitudes statistically. Across a roadmap of configurations, we perform diagnostic ablations showing that the gains depend primarily on the discriminator redesign. We train all generators and discriminators from scratch, without pretrained discriminator features. The resulting R3GAN2 models reach FIDs of 2.12 with 17 M parameters, 1.60 with 61 M, and 1.52 with 131 M on class-conditional ImageNet-256. At 1 NFE, this is substantially more efficient than competing diffusion and autoregressive methods, and than prior GAN baselines too (e.g., 10× fewer parameters than StyleGAN-XL for similar FID). We also train directly in pixel space, achieving FIDs of 1.65 on CIFAR-10 and 2.19 on FFHQ 256×256, where our generator achieves comparable FID to StyleGAN-XL with fewer than half as many parameters. In sum, identifying and removing this discriminator magnitude pathology yields a new SOTA in FID/parameter efficiency for GANs.

1 Introduction

A focus of generative modeling is transforming one distribution into another, typically from a tractable distribution such as an isotropic Gaussian into an unknown target distribution that we seek to reconstruct from a set of sample points. Variational autoencoders (VAE) [1], normalizing flow (NF) [2], and generative adversarial networks (GAN) [3, 4] perform such a transformation in one step. Other processes decompose the transformation iteratively, such as autoregression (AR) [5–7], diffusion [8–10], or flow matching [11] models, and can achieve high sample quality. However, iteration is computationally expensive, and so there is interest to revive one-step generative models [12, 13].

We consider GANs; specifically, our modern R3GAN baseline [14], which showed that GAN training can be stable with a relativistic loss [15, 16] and improved regularizers [17]. However, R3GAN was evaluated on low-resolution datasets like CIFAR [18] and homogeneous datasets like faces, and the question of whether this approach can deliver competitive results on diverse datasets at higher resolution remains unanswered. Indeed, our experiments on ImageNet 256×256 [19] expose additional training-dynamics pathologies and, despite stable training, R3GAN fails to achieve satisfactory sample quality as FID plateaus early in training. Inspecting the discriminator reveals a striking internal pathology accompanying this saturation: its activation magnitudes grow persistently by many orders through training, particularly in the late layers, while generator magnitudes remain comparatively small.

Karras *et al*. noted a related magnitude pathology in the training of diffusion models [10]. In a GAN discriminator, activation scale has an additional connection to the learning objective: the discriminator is trained directly to separate real and generated samples by their scalar scores. Once some useful

40th Conference on Neural Information Processing Systems (NeurIPS 2026).
*Equal contribution.

discriminative responses have been learned, the real–fake separation can in principle be increased by amplifying the existing responses without necessarily learning new discriminative structure. This creates a simple scale degree of freedom that motivates our investigation of the observed magnitude growth. In Sec. 2, we examine this possibility more carefully and show that such amplification is not ruled out by the regularized empirical discriminator objective. Then, we ask whether removing uncontrolled activation-scale growth improves discriminator learning in practice.

To test whether controlling this scale degree of freedom resolves the pathology, we take an architecture-wide magnitude-preserving approach that is inspired by recent work in manifold-constrained optimization [10, 20–22]. First, we explain how each architecture component helps to preserve magnitude statistically. Then, we use an ablation roadmap to show that our redesigned R3GAN2 model removes the observed magnitude growth and improves FID on ImageNet 256 × 256 from 28.67 to 2.56. Ablations show that the gains from magnitude preservation depend primarily upon the discriminator redesign.

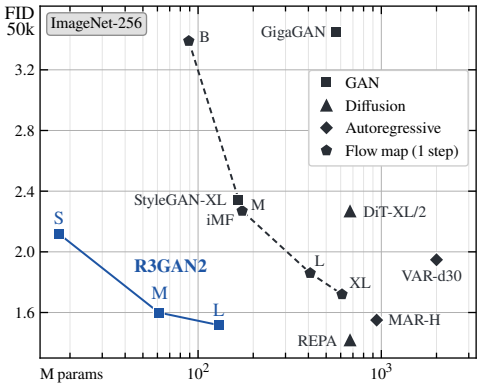


Figure 1: Magnitude-preserving layers create efficient GANs with low FID without guidance.

We train all generators and discriminators from scratch, without pretrained discriminator features [23, 24]. Our smallest final model achieves an FID of 2.12 with 17 M parameters at 1 NFE, compared with DiT-XL/2 [25], which achieves 2.27 FID with 250×2 NFE (classifier-free guidance) using 675 M parameters—an estimated 190× more inference FLOPs (including autoencoder decoding for both models). Increasing the generator to 131 M parameters further improves ImageNet FID to 1.52. We also train directly in pixel space, achieving FIDs of 1.65 on CIFAR-10 and 2.19 on FFHQ 256 × 256, where our 27 M-parameter generator achieves comparable FID to StyleGAN-XL [24] with fewer than half as many parameters. Overall, R3GAN2 shows a significant improvement in the FID/parameter count trade-off (Fig. 1), further uncovering the potential of GANs.

2 Magnitude Growth in Adversarial Training

R3GAN. We begin by reviewing the adversarial objective and regularization used by R3GAN. R3GAN adopts an alternative adversarial objective named RpGAN [15, 16]. Given training samples $x \sim p_{\mathcal{D}}$ and a random variable z sampled from a prior distribution p_z, the generator G_θ parameterized by θ and the discriminator D_ψ parameterized by ψ seek to optimize the following objective:

$$\mathcal{L}(\theta, \psi) = \mathbb{E}_{\substack{z \sim p_z \\ x \sim p_{\mathcal{D}}}} \left[f\big(D_\psi(x) - D_\psi(G_\theta(z))\big) \right] \tag{1}$$

G is optimized to minimize $\mathcal{L}$ while D tries to maximize it. A common choice of f is $f(t) = -\log(1 + e^{-t})$ which recovers the relativistic version [15] of the classic GAN [3].

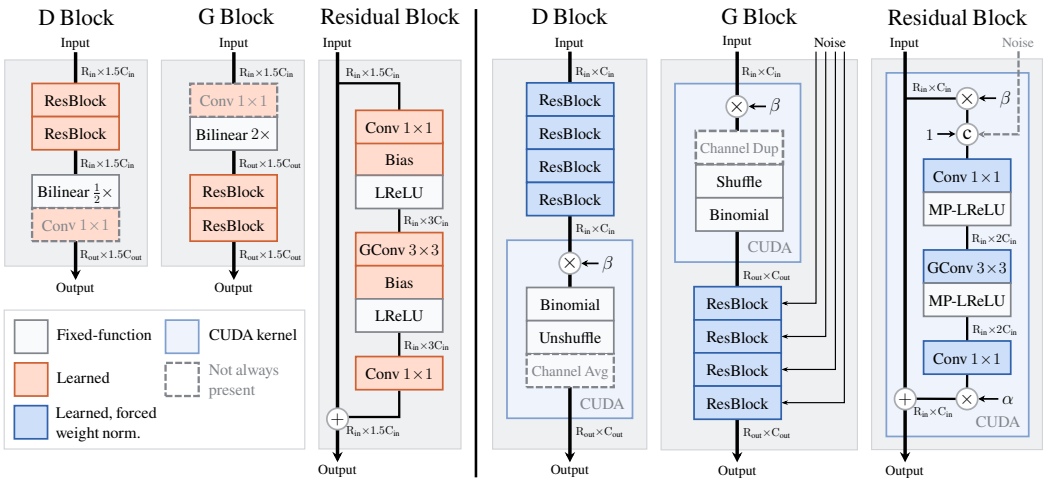
Sun *et al*. [16] prove that while the classic GAN loss is well-behaved in probability space, the empirical loss contains exponentially many mode-collapse local minima. They show that the RpGAN loss landscape is not plagued by the same pathology and every basin is a global minimum. This is an important step to understand the empirical training dynamics of GANs, but mode collapse is not our only challenge. On ImageNet, we observe a distinct pathology in the internal magnitudes of the discriminator despite otherwise stable adversarial training.

For convergent training, R3GAN [14] extends the work of Mescheder *et al*. [17] to Eq. 1, which leads to two additional regularization terms R_1 and R_2 applied when optimizing D:

$$R_1(\psi) = \frac{\gamma}{2}\mathbb{E}_{x \sim p_{\mathcal{D}}} \left[\|\nabla_x D_\psi\|^2 \right] \qquad R_2(\theta, \psi) = \frac{\gamma}{2}\mathbb{E}_{x \sim p_\theta} \left[\|\nabla_x D_\psi\|^2 \right] \tag{2}$$

R_1 penalizes the gradient norm of D on real samples, and R_2 similarly on fake samples. Importantly, these penalties constrain the input–output Jacobian of the discriminator at sampled real and generated points; they do not directly constrain the scale of its internal representations. Therefore, whether they also control internal activation magnitude depends on how a particular discriminator realizes its input–output function.

Magnitude pathology. Despite stable adversarial optimization, we observe that discriminator activation magnitudes grow continuously through ImageNet training, with the strongest growth concentrated in late layers; in contrast, generator magnitudes remain comparatively small.



(a) R3GAN architecture blocks (CONFIG A) (b) R3GAN2 architecture blocks (CONFIG D)

Figure 2: **Architecture comparison.** Both R3GAN and R3GAN2 use approximately symmetric convolutional architectures for G and D, organized into stages of residual blocks separated by resolution transitions. (a) R3GAN removed StyleGAN2's style mapping and modulation machinery, while modernizing the architecture with a cleaner ConvNet design. (b) R3GAN2 retains this clean convolutional design while updating the architecture to statistically preserve signal magnitude. We replace learned projection-based resolution changes with non-parametric shuffle-based resampling and reparameterize the residual blocks to control variance drift. Stabilizing discriminator magnitudes enables a deeper-and-narrower design with four residual blocks per stage instead of two.

To reason about how such growth can arise, let us write the discriminator locally as $D(x) = g(z(x))$ to separate changes in what is represented for each input from changes in how strongly already-represented differences affect the discriminator score: $z(x)$ is the input-specific score representation, while g is the shared downstream mapping from that representation to the scalar score. Once $z(x)$ contains directions that distinguish real and generated samples, their score separation can increase either through changes to the represented discriminative structure or through increased sensitivity of g along directions that are already discriminative. The latter provides a scale degree of freedom: existing discriminative responses can be amplified without requiring a corresponding change in the structure represented by $z(x)$.

We examine this decomposition in two complementary ways. 1) Mathematically, in App. A.2, we construct an exactly analyzable limiting case in which $z(x)$ is a one-dimensional separating score. Then, increasing the gain of g amplifies both internal magnitude and empirical real–fake separation without changing the underlying separator; notably, empirical R_1 and R_2 do not prevent this behavior. 2) Empirically, we show that our trained discriminators exhibit consistent behavior with the same broader picture: persistent real–fake response directions can remain useful across training while their amplitudes increase substantially as discriminator magnitudes grow (App. A.3). Together, these observations indicate that pathological magnitude growth is accompanied by repeated amplification of already-discriminative structure. This motivates treating magnitude preservation as an architecture-wide design objective, which we develop next.

3 Treating the Magnitude Pathology

The most direct way to prevent magnitude growth is to explicitly normalize intermediate activations [26, 27]. However, in our experiments, this severely degrades sample quality, consistent with prior generative models where direct feature normalization has also proven undesirable [10, 14, 28, 29]. Therefore, we take a less restrictive approach [10, 28]: rather than fixing the magnitude of each activation explicitly, we design individual network components to preserve magnitude statistically while retaining input-dependent variation. These form an architecture (Fig. 2).

Weight normalization. First, we consider linear layers, which make up most of the learned transformations in the network and provide a direct route for changing activation scale. For a linear map, magnitude preservation can be expressed as a constraint on its weights, but the appropriate constraint depends on what we assume about the input statistics. For an i.i.d. input, amplification of each output is governed by the row-wise ℓ_2 norm; under worst-case channel correlation, the corresponding quantity is the row-wise ℓ_1 norm; and if we instead constrain the worst-case ℓ_2 amplification of the

entire linear map, the relevant quantity is its spectral norm. We adopt the standard i.i.d. mean-field assumption [10, 30–32], under which row-wise ℓ_2 normalization preserves activation variance [10]. Concretely, let $\mathbf{x}$ be a random vector where each component x_i is i.i.d. and has a variance of σ_x^2. Let $\mathbf{w}$ be a unit vector and $y = \mathbf{w} \cdot \mathbf{x}$, the standard deviation σ_y is given by

$$\sigma_y = \sqrt{\mathrm{Var}[\mathbf{w} \cdot \mathbf{x}]} = \sqrt{\sum_i w_i^2 \mathrm{Var}[x_i]} = \sqrt{\sum_i w_i^2 \sigma_x^2} = \|\mathbf{w}\|_2 \, \sigma_x = \sigma_x \tag{3}$$

As Eq. 3 relies on an i.i.d. input assumption, it is not a hard constraint on activation magnitude. Amplification remains possible when the inputs become correlated. In particular, row-wise ℓ_2 normalization fixes the squared Frobenius norm of the weight matrix—the sum of its squared singular values—but does not fix the individual singular values. Thus, a few singular values can grow while others shrink, concentrating the weight energy into a smaller number of dominant directions; we refer to this behavior as *spectral sharpening*. Inputs aligned with these directions can then be amplified despite the fixed row-wise ℓ_2 norms. In practice, we find this residual degree of freedom benign in typical runs; substantial spectral sharpening appears mainly in the degenerate settings examined in Sec. 4.1, where we also examine the stronger ℓ_1 and spectral constraints. Further discussions of these alternative constraints are given in App. B.1.

Activation function. Controlling linear layers alone is insufficient, since nonlinearities also change activation statistics. Following Brock *et al*. [31, 32], we use the same mean-field principle as above: assume approximately i.i.d. inputs and scale the nonlinearity so that its output variance matches its input variance. Consider the leaky ReLU activation function used in R3GAN. Under the common Gaussian pre-activation assumption, $x \sim \mathcal{N}(0, \sigma_x^2)$, the variance after LReLU with leak α admits a closed form [33]:

$$\mathrm{Var}[\mathrm{LReLU}_\alpha(x)] = \frac{\sigma_x^2}{2}\left((1+\alpha^2) - \frac{(1-\alpha)^2}{\pi}\right) \tag{4}$$

Applying this result layer-by-layer requires controlling the pre-activation mean. The output variance of LReLU depends on both the input variance and mean, while LReLU itself induces a positive mean shift. Therefore, activations can drift away from zero mean with depth, making variance propagation difficult to predict from variance alone.

A cleaner approach is to enforce zero-mean pre-activations before LReLU. We achieve this by *weight centering*: constraining each row of the preceding linear layer to have zero sum [31, 32]. Suppose the pre-activation is $y = \mathbf{w} \cdot \mathbf{x}$, where the input vector $\mathbf{x}$ has i.i.d. components with mean μ_x. If $\mathbf{w}$ is zero-centered in the sense that $\sum_i w_i = 0$, then the pre-activation is zero-mean regardless of μ_x:

$$\mu_y = \mathbb{E}[\mathbf{w} \cdot \mathbf{x}] = \sum_i w_i \mathbb{E}[x_i] = \mu_x \sum_i w_i = 0 \tag{5}$$

This guarantees a zero-mean pre-activation independent of the input mean. Under the same mean-field assumption [30] that pre-activations are approximately i.i.d. Gaussian, Eq. 4 now applies layer-by-layer, making variance propagation through the nonlinearity tractable.

Residual connections. Consider a stack of n residual blocks [34] where f_i denotes the i-th residual branch and x_i the block output: $x_i = x_{i-1} + f_i(x_{i-1})$. Even if the primitives inside f_i (linear layers, nonlinearities) are individually magnitude preserving, the *addition itself* introduces variance drift. Under the same simplifying i.i.d. assumption, namely that the skip term x_{i-1} and the residual term $f_i(x_{i-1})$ are approximately uncorrelated with comparable variance, we obtain $\mathrm{Var}[x_i] \approx 2\mathrm{Var}[x_{i-1}]$ and thus $\mathrm{Var}[x_n] \approx 2^n \mathrm{Var}[x_0]$. This implies exponential growth in depth if left unchecked. We might renormalize after every residual addition [10, 28], *e.g.*: $x_i = (x_{i-1} + f_i(x_{i-1}))/\sqrt{2}$, but this attenuates the identity path at every block and compromises the identity-preserving behavior of pre-activation ResNets [35]. Instead, we control the strength of each residual branch while leaving the skip path unchanged. We parameterize each residual update with an explicit learnable strength α_i, and scale the input to the residual branch by a deterministic factor β_i [31, 32]:

$$x_i = x_{i-1} + \alpha_i f_i(\beta_i x_{i-1})\,. \tag{6}$$

The role of β_i is to keep the residual branch operating on a stable input scale, mimicking the effect of the pre-activation normalization in ResNet v2 [35], so that variance does not drift as residual contributions accumulate. Under the same i.i.d./uncorrelated-branch approximation, this can be achieved by choosing β_i as a function of the previous residual strengths:

$$\beta_1 = 1 \qquad \beta_i = \frac{1}{\sqrt{1 + \sum_{j=1}^{i-1} \alpha_j^2}}\,. \tag{7}$$

By design, β_i is chosen so that $\mathrm{Var}[\beta_i x_{i-1}] \approx \mathrm{Var}[x_0]$ for all i. In particular, the final rescaling ensures $\mathrm{Var}[\beta_{n+1} x_n] \approx \mathrm{Var}[x_0]$, so the network output retains the same variance scale as the input.

4 A Roadmap of Improvements

Starting from R3GAN [14], we introduce architectural changes incrementally and track their effects on sample quality and discriminator magnitude (Tab. 1). Unless otherwise noted, we evaluate all configurations on class-conditional ImageNet 256×256. As is standard for high-resolution generation [6, 10, 12, 13, 25, 36], our GAN operates in a latent space using the fixed, open-source FLUX.2 autoencoder [37], which provides $8\times$ spatial downsampling to a $32 \times 32 \times 32$ latent tensor and decodes generated latents back to image space. On ImageNet, all roadmap configurations are trained for 1000 M images with batch size 4096. We additionally use CIFAR-10 in pixel space for selected diagnostic ablations, trained for 100 M images with batch size 512.

Configurations / ImageNet 256×256	FID↓	Params M, G / D
A R3GAN [14]	28.67	20.87 / 22.06
B + Preliminary improvements	19.20	17.03 / 18.21
B* + Premature depth scaling	37.43	16.30 / 16.93
C + Magnitude-preserving architecture	5.37	19.09 / 20.76
D + Depth scaling revisited	**2.56**	17.41 / 18.24

Table 1: While preliminary improvements appear to improve performance, deepening the network in CONFIG B* degrades performance and discriminator magnitudes continue to grow. Our magnitude-preserving architecture improves FID in CONFIG C and enables a deeper-and-narrower network in CONFIG D to achieve low FID efficiently.

Baseline (CONFIG A). We adapt R3GAN's ImageNet 32×32 configuration to our experimental setting with two changes: (1) increasing the generator output and discriminator input from 3 to 32 channels to match the autoencoder latent space, and (2) reducing model width by $2\times$ for faster iteration. This yields approximately 20 M parameters each for G and D. Following Karras *et al.* [10], we globally standardize the training-set latents to zero mean and standard deviation 0.5. CONFIG A obtains an FID of 28.67. Figure 3 shows a persistent increase in discriminator activation magnitude throughout training, concentrated most strongly in the late 4×4 stage near the classifier. This reproduces the magnitude pathology studied in Sec. 2 within our ImageNet training setting.

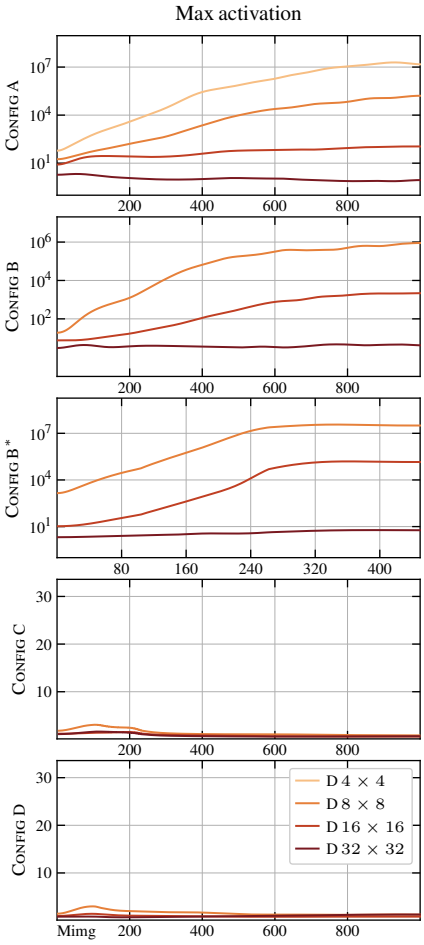


Figure 3: Max activation per stage of the discriminator (App. A.1).

Preliminary improvements (CONFIG B). First, we apply several architectural and optimization improvements unrelated to magnitude preservation (App. C.1). We increase the group size of the 3×3 convolutions from 16 to 32, remove the 4×4 residual blocks [9], adopt power EMA for post-hoc tuning [10], simplify the bias parameterization [14], replace 1×1 + bilinear resampling with shuffle-based resampling [38] and low-pass filtering [39], and reintroduce noise injection [39, 40] in the generator. Together, these changes improve FID from 28.67 to 19.20. However, discriminator activation magnitudes continue to grow substantially throughout training (Fig. 3); the improved baseline retains the same magnitude pathology.

Premature depth scaling (CONFIG B*). Second, we reallocate capacity from width to depth while keeping the parameter count approximately fixed, doubling the number of blocks per stage from 2 to 4 and reducing width from 768 to 512. This deeper-and-narrower configuration initially appears promising, with FID decreasing substantially faster than CONFIG B early in training. However, the improvement does not persist: FID eventually plateaus at 37.43, while peak discriminator activation magnitude increases from approximately 10^6 to 10^8. At this scale, training also becomes increasingly sensitive to floating-point errors. We revisit the same deeper-and-narrower design in CONFIG D after addressing the magnitude pathology in CONFIG C.

Magnitude-preserving architecture (CONFIG C). Next, we directly address the discriminator magnitude pathology by applying the magnitude-preserving components from Sec. 3 to D: row-wise ℓ_2 weight normalization [10], weight centering [32] and magnitude-preserving LReLU [33], and magnitude-preserving residual connections [31, 32]. CONFIG C also adds feed-forward heads and EDM2's forced weight normalization and learning-rate schedule [10]. Table 2 adds these changes one at a time: ℓ_2 weight normalization alone gives most of the gain, and together they lower FID from 19.20 to 5.37 (details in App. C.2). With CONFIG C, the discriminator magnitude growth disappears: peak activations stay below about 4.2 throughout training rather than growing by orders of magnitude (Fig. 3).

Although G does not exhibit the same magnitude pathology (App. A.1), we use the same magnitude-preserving parameterization in G in the final CONFIG C. This provides an additional empirical gain when combined with MP-D and simplifies hyperparameter tuning by keeping the two networks under a shared optimization regime. Magnitude-preserving weight parameterizations substantially alter optimization scale and learning-rate requirements [10]; applying them symmetrically avoids introducing separate optimization regimes for G and D. Consistent with the discriminator-side pathology, applying the magnitude-preserving parameterization to G alone provides no improvement over CONFIG B, whereas applying it to D alone yields most of the FID reduction (Tab. 3).

Cumulative, in G_θ & D_ψ	FID↓
CONFIG B	19.20
+ ℓ_2 weight normalization	9.82
+ MP LReLU and weight centering	9.71
+ MP residuals	9.29
+ Feed-forward heads	8.07
+ Forced WN, EDM2 LR (CONFIG C)	**5.37**

Table 2: CONFIG C ablation.

Magnitude-preserving architecture	FID↓
Neither (CONFIG B)	19.20
G_θ only	20.18
D_ψ only	8.68
G_θ & D_ψ (CONFIG C)	**5.37**

Table 3: **MP in D, not G, drives the gain.**

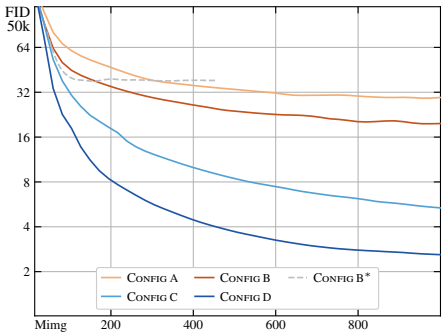


Figure 4: FID-50k across all configs.

Depth scaling revisited (CONFIG D). With discriminator magnitudes stabilized in CONFIG C, we revisit the same deeper-and-narrower architecture as CONFIG B*. In contrast to CONFIG B*, discriminator activations remain bounded throughout training, and the promising early FID trajectory now translates into a final FID of 2.56 (Fig. 4).

4.1 Diagnostic Ablations

Table 4 compares several alternatives to our magnitude-preserving design. Along with FID and peak discriminator activation magnitude, we report the squared input-gradient norms on real samples, $\|\nabla_{x_\mathcal{D}} D\|^2$, and on generated samples, $\|\nabla_{x_\theta} D\|^2$, which are the quantities entering R_1 and R_2. We also track the discriminator ranking statistic $r_t = \mathbb{E}[\mathrm{sgn}(D(x_\mathcal{D}) - D(x_\theta))]$, where $r_t = 1$ indicates that D consistently ranks real samples above generated samples, $r_t = 0$ indicates no consistent ordering, and $r_t = -1$ indicates reverse ordering [41]. λ denotes the explicit discriminator output gain noted below.

Table 4: **Diagnostic ablations.** Discriminator output gain is slightly worse, alt. weight constraints is significantly worse, and self-attention is slightly worse. Peak mag.: the largest per-stage max activation (App. A.1).

Variant	Peak mag.	λ	$\|\nabla_{x_\mathcal{D}} D\|^2$	$\|\nabla_{x_\theta} D\|^2$	r_t	FID↓
CIFAR-10						
CONFIG D (baseline)	5.68	1	3.55	3.53	0.386	**1.81**
Discriminator gain						
+ Learnable λ	337	0.003	3.52	3.51	0.323	1.93
+ $\nabla_\lambda[R_1 + R_2]$	0.149	224.9	4.63	4.57	0.348	1.91
Alternative weight constraints						
ℓ_1 weight norm	0.228	1	3.6×10^{-6}	3.6×10^{-6}	0.013	166.29
Spectral norm	0.118	1	1.3×10^{-3}	1.2×10^{-3}	0.042	31.12
Self-attention						
+ Attention	3.11	1	8.69	8.06	0.622	2.03
ImageNet 256 × 256						
CONFIG D (baseline)	1.38	1	0.141	0.133	0.616	**2.56**
+ Attention	2.99	1	0.182	0.175	0.794	2.76

Explicit discriminator gain. Our analysis in Sec. 2 suggests that part of the discriminator's progress can come from increasing sensitivity along already-discriminative directions. Therefore, a natural alternative to allowing this scale change to emerge inside the network is to expose it explicitly at the output: $D_{\psi,\lambda}(x) = \lambda D_\psi(x)$. In principle, the learnable gain λ can absorb changes in output scale while reducing the incentive to realize the same amplification through the discriminator trunk. Naively, this does not work: because substituting $D_{\psi,\lambda}$ into Eq. 2 makes both R_1 and R_2 scale as λ^2, λ collapses toward zero, while the discriminator compensates through spectral sharpening and the magnitude pathology re-emerges in the trunk (Fig. 5). Stopping the penalty gradient ($\not\nabla$) through λ reverses this behavior: λ grows large and trunk activations remain small, but sample quality does not improve over CONFIG D (Tab. 4). We analyze these two outcomes in detail in App. D.1.

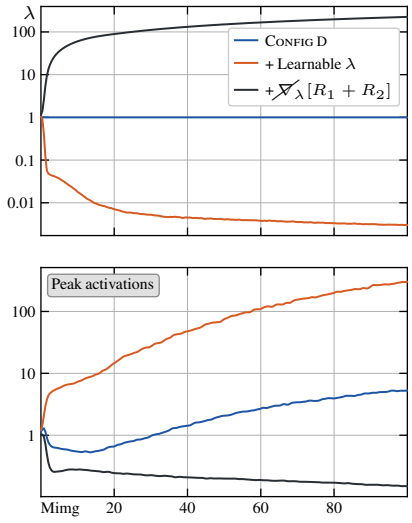


Figure 5: Discriminator gains.

Alternative weight constraints. Row-wise ℓ_2 normalization preserves magnitude statistically under the i.i.d. assumptions of Sec. 3 and still permits spectral sharpening. Therefore, we test stronger row-wise ℓ_1 and spectral normalization [42] constraints that suppress this remaining amplification route. In both cases, discriminator magnitudes decrease throughout training, accompanied by r_t approaching zero and very small discriminator input gradients (Tab. 4 and Fig. 6). These runs perform substantially worse than CONFIG D, consistent with the discriminator becoming too contractive to discriminate effectively. Detailed discussion is provided in App. D.2.

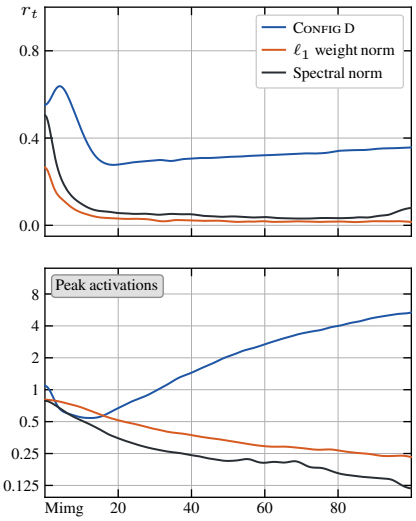


Figure 6: Alternative weight constraints.

Self-attention. We also test whether the magnitude-preserving design extends to self-attention by inserting a single attention block into both G and D. We adapt the attention layer to preserve magnitude (App. D.3). With this design, attention trains stably without runaway magnitude growth and reduces FID faster early in training; it plateaus sooner, ending close to CONFIG D on both CIFAR-10 and ImageNet (Fig. 7). The higher r_t and discriminator input gradients indicate higher discriminative accuracy and greater input sensitivity, suggesting overfitting (Tab. 4). This is consistent with prior observations that vision transformers often require stronger regularization and larger data regimes [43]. Although we do not use attention in our final models, a well-behaved magnitude-preserving attention block provides a basis for future architectures that require global interactions [44] or conditioning [5, 45].

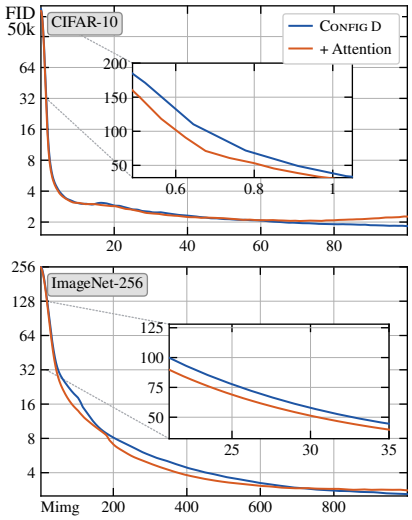


Figure 7: Self-attention FID progression.

5 Results

Following the analyses in Sec. 4, we evaluate image quality, distribution coverage, and generation cost on class-conditional ImageNet 256×256. To assess whether the benefit of magnitude preservation generalizes beyond ImageNet, we evaluate in pixel space on CIFAR-10 and FFHQ 256×256. In every setting, the generator and discriminator are optimized from scratch, without pretrained discriminator features or external feature-alignment objectives. Our central result is that compact single-step GANs obtain strong fidelity alongside improving coverage as capacity grows. We report FID [46, 47] and $\text{FD}_{\text{DINOv2}}$ [48]; Apps. E.1 and F.1 give the training details and protocol.

Table 5: **R3GAN2 efficiently reaches low FID without guidance.** Class-conditional ImageNet 256×256; Tab. F.2 adds more methods and standard deviations.

- **FD**: 50k samples; mean of five seeds; † published under paper's protocol; '–': none or not reported.
- **Feat.**: additional pretrained features in training, which can flatter that family's FD [49]. **I**: supervised ImageNet classifier, like FID's Inception; **D**: DINO family, like $\text{FD}_{\text{DINOv2}}$'s. AE training (Domain) may also use them.
- **Guide**: guidance reshapes the samples' distribution away from the model's learned conditional distribution. **CFG**: classifier-free guidance; **AG**: AutoGuidance; **CG**: classifier guidance from a pretrained classifier; subscript **t**: trained in, no sampling cost. TFLOPs include guidance.

								FD↓, unguided		FD↓, guided	
Method	Type	Params	Domain	Feat.	Guide	NFE	TFLOPs	Incept.	DINOv2	Incept.	DINOv2
MAR-H [6]	AR	943 M	KL-16	–	CFG	256×2	≈189.52	2.35†	–	1.55†	–
RAR-XXL [50]	AR	1,500 M	MaskGIT-VQ	–	CFG	256×2	≈ 1.73	–	–	1.49	160.9
JLT-B/1 [51]	Diffusion	130 M	FLUX.2 AE	–	CFG	99×2	≈ 9.75	16.12	367.3	2.96	160.9
DiT-XL/2 [25]	Diffusion	675 M	SD-VAE	–	CFG	250×2	≈119.12	9.62†	–	2.27†	–
SiT-XL/2+REPA [52, 53]	Diffusion	675 M	SD-VAE	D	CFG	250×2	≈100.64	5.90†	–	1.42†	–
RAE [54]	Diffusion	839 M	DINO-B	–	AG	50×2	≈ 18.00	1.51	48.0	1.15	35.9
α-Flow-XL/2+ [55]	Flow-map	676 M	SD-VAE	–	CFG_t	1	≈ 0.86	–	–	2.46	152.7
iMF-XL/2 [56]	Flow-map	610 M	SD-VAE	–	CFG_t	1	≈ 0.97	10.61	272.3	1.82	105.4
					CFG_t	2	≈ 1.32	–	–	1.61	91.9
FACM-XL/2 [57]	Flow-map (distilled)	675 M	VA-VAE	–	CFG_t	1	≈ 0.86	–	–	1.72	82.4
					CFG_t	2	≈ 1.10	–	–	1.34	65.4
StyleGAN-XL [24]	GAN	166 M	Pixel	I	CG_t	1	3.14	–	–	2.34	214.9
StyleSAN-XL [58]	GAN	166 M	Pixel	I	CG_t	1	3.14	–	–	2.27	224.1
R3GAN2-S (ours)	GAN	17 M	FLUX.2 AE	–	–	1	0.64	2.12	163.9	–	–
R3GAN2-M (ours)	GAN	61 M	FLUX.2 AE	–	–	1	0.67	1.60	125.7	–	–
R3GAN2-L (ours)	GAN	131 M	FLUX.2 AE	–	–	1	0.73	1.52	115.1	–	–

5.1 ImageNet 256×256: Quality, Coverage, and Efficiency

Image quality and scaling. ImageNet has long posed a challenge for GANs. BigGAN [59] demonstrated high-fidelity synthesis but remained susceptible to training collapse and limited distribution coverage, while the StyleGAN [28, 39, 60] family struggled to extend its success to this diverse dataset. StyleGAN-XL [24] achieved strong ImageNet results through a combination of pretrained convolutional and transformer feature networks, multiple projected discriminators [23], progressive growing [61], and auxiliary classifier guidance [47]. R3GAN2 achieves better FID with a simpler adversarial setup: a single convolutional generator–discriminator pair trained jointly at a fixed resolution, without pretrained discriminator features or auxiliary classifier guidance. R3GAN2-S reaches FID 2.12 with 17 M, about one-tenth of StyleGAN-XL's parameter count. Increasing architecture width further improves FID. R3GAN2-M also achieves lower FID than contemporary single-step generators iMF-XL/2 [56] and FACM-XL/2 [57], using approximately one-tenth their generator parameters. Without guidance, our model at all three sizes outperforms diffusion methods DiT-XL/2 [25] and SiT-XL/2+REPA [52, 53], and R3GAN2 remains competitive against guided diffusion. For DINOv2 FD, R3GAN2 also performs better than StyleGAN-XL. Compared to diffusion and one-step generators, R3GAN2 is less often competitive though continues to show improvements as model scale increases (samples in Fig. 8).

Distribution coverage. Limited distribution coverage has long been a concern for GANs [47], with BigGAN-deep [59] obtaining recall of 0.278 and StyleGAN-XL [24] of 0.496. R3GAN2-S reaches recall of 0.557, increasing to 0.596 for R3GAN2-L (Tab. 6). These coverage gains accompany lower FID while precision remains 0.81–0.82 across scales, with R3GAN2-S matching StyleGAN-XL's precision (0.810 versus 0.809). The joint improvement in fidelity and coverage shows that the limited coverage of earlier GANs is not intrinsic to adversarial training. The resulting precision–recall balance is comparable to that of representative guided diffusion models, despite R3GAN2 using no guidance: ADM-G (0.52), LDM-4-G (0.55; guidance scale 1.25), and DiT-XL/2-G (0.57; guidance scale 1.50), with similarly high precision of 0.81–0.83 [25, 47, 63]. Contemporary guided one-step flow maps lean the other way, with higher recall (0.61–0.63) and lower precision (0.77–0.78); without guidance, iMF-XL/2 moves further that way.

Table 6: R3GAN2 matches diffusion's precision, with higher recall than earlier GANs. ImageNet 256×256.

Method	Guide	FID↓	Prec.↑	Recall↑
BigGAN-deep [59]	–	6.89	0.872	0.278
StyleGAN-XL [24]	CG_t	2.34	0.809	0.496
R3GAN2-S (ours)	–	2.12	0.810	0.557
R3GAN2-M (ours)	–	1.60	0.820	0.590
R3GAN2-L (ours)	–	1.52	0.818	0.596
α-Flow-XL/2+ [55]	CFG_t	2.46	0.765	0.614
iMF-XL/2 [56]	CFG_t	1.82	0.782	0.625
iMF-XL/2, ω=1 [56]	–	10.61	0.624	0.698
FACM-XL/2 [57]	CFG_t	1.72	0.783	0.633
ADM-G [47]	CG	4.59†	0.82†	0.52†
LDM-4-G, s=1.25 [63]	CFG	3.95†	0.81†	0.55†
DiT-XL/2-G, s=1.50 [25]	CFG	2.27†	0.83†	0.57†

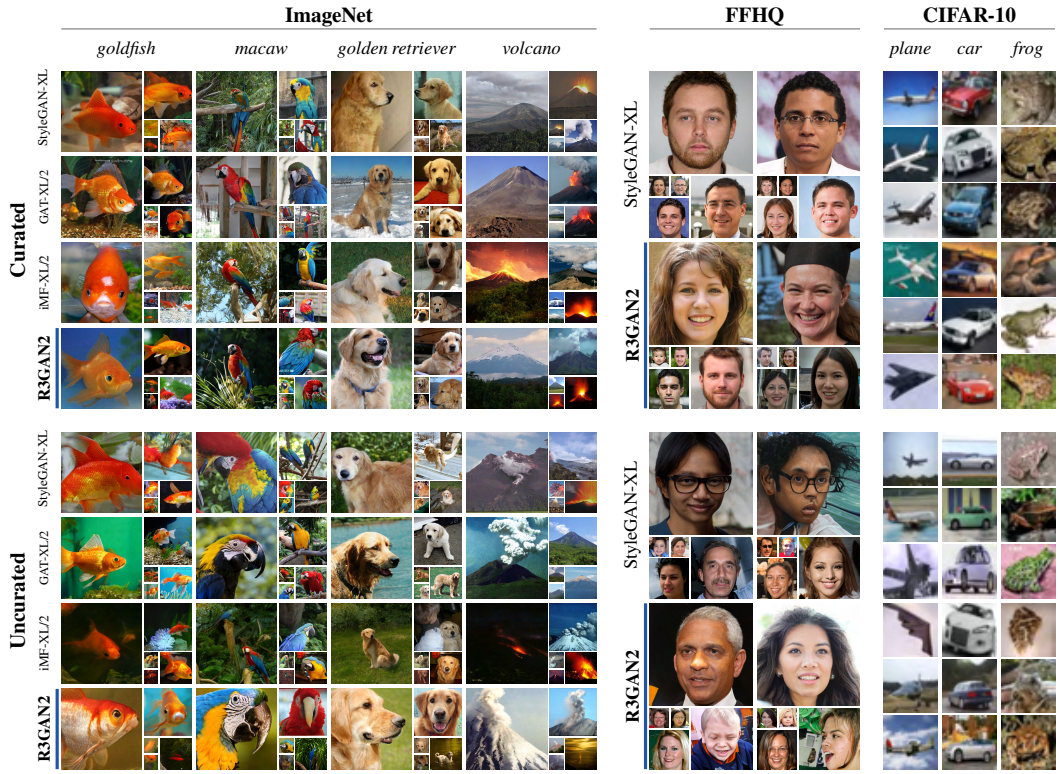


Figure 8: Curated (top) and uncurated (bottom) samples from StyleGAN-XL [24], GAT-XL/2 [62], iMF-XL/2 [56] and R3GAN2. StyleGAN-XL trains with pretrained ImageNet classifiers and classifier guidance, GAT-XL/2 aligns to DINOv2 features, and iMF-XL/2 trains classifier-free guidance into the model. R3GAN2 uses neither but produces comparable quality.

Generation efficiency. To generate a sample, guided DiT-XL/2 [25] requires 120 TFLOPs over 250 sampling steps with two evaluations per step for classifier-free guidance. One R3GAN2 sample requires one generator evaluation and one autoencoder decode: R3GAN2-S uses 0.64 TFLOPs, with M at 0.67 and L at 0.73, with AE decode included at 0.62 TFLOPs. This is 190× less than DiT-XL/2. Single-step models are more efficient than diffusion, and comparable to R3GAN2: iMF-XL/2 [56] uses 0.97 TFLOPs and FACM-XL/2 [57] uses 0.86 TFLOPs. With a fixed and dominant decoding overhead, scaling R3GAN2 from S to L improves both FID and recall for little cost.

5.2 Pixel-Space Synthesis: FFHQ 256×256 and CIFAR-10

For FFHQ synthesis, we use CONFIG C rather than CONFIG D, as we find empirically that this dataset favors a wider, shallower architecture. Our 27 M-parameter generator achieves FID 2.19 (Tab. F.4), improving on R3GAN's 2.76 [14]. This is comparable to StyleGAN-XL's 2.19 [24], while using fewer than half its 68 M generator parameters and approximately 7× fewer TFLOPs per image (0.18 versus 1.28). We obtain these results directly in pixel space without an autoencoder or pretrained feature supervision in either network, whereas StyleGAN-XL uses pretrained ImageNet features in its discriminator. On class-conditional CIFAR-10, R3GAN2 achieves FID 1.65 (Tab. F.3) with 17 M generator parameters, improving on R3GAN's 2.00 [14]. It also improves on class-conditional EDM [9] (1.81), using a single network function evaluation compared with 35 NFEs.

5.3 Post-hoc EMA Sensitivity

We use EDM2's post-hoc EMA reconstruction [10] to evaluate image quality across averaging horizons without retraining (Fig. 9). The released models use σ_{rel} of 8.5% (S), 7% (M) and 9.5% (L) on ImageNet, 4.5% on CIFAR-10 and 3% on FFHQ. Sensitivity varies across datasets: ImageNet and CIFAR-10 exhibit broad near-optimal regions, although very short horizons substantially degrade ImageNet quality. FFHQ is particularly insensitive: from 1% to 9%, FID varies less than between runs, and it rises only beyond about 10%. Despite these differences, all of our models are less sensitive to the precise averaging horizon than EDM2, which exhibits a sharply defined optimum. The origin of this difference in averaging behavior remains unclear.

6 Discussion, Limitations, & Future Work

Learning without pretrained feature supervision. R3GAN2 does not use pretrained feature backbones or external representation-alignment targets in either network. This differs from the pretrained discriminator features used by StyleGAN-XL [24] and the alignment targets used by REPA [53]. Although this simplifies the adversarial training setup, it requires solving a harder joint learning problem: learning to generate while learning representations that define a meaningful measure of distributional discrepancy in high-dimensional space. Drifting Models [13] illustrates this difficulty: its authors report unsuccessful ImageNet training without a feature encoder, even with VAE latents, and instead compute sample similarities in pretrained feature spaces. For ImageNet, R3GAN2 uses the FLUX.2 AE latent space, whose training data and procedure are not public. However, R3GAN2's performance is not explained by the FLUX.2 AE latent space: the JLT-B/1 diffusion model with 130 M parameters, trained on the same latents, produces worse unguided results than our parameter-matched L model (FID 1.52 vs. 16.12; DINOv2 FD 115.1 vs. 367.3). Improving the ability to learn representations within the generative training process is motivated by transfer-learning studies: on COCO [64], He *et al.* [65] show that longer training allows randomly initialized models to match pretrained counterparts, while Zoph *et al.* [66] find that more target-task data and stronger augmentation can diminish the benefits of pretraining. These findings motivate strengthening direct task learning, and our results demonstrate progress toward this goal by developing effective discriminator representations within joint adversarial optimization. This provides a foundation for further scaling.

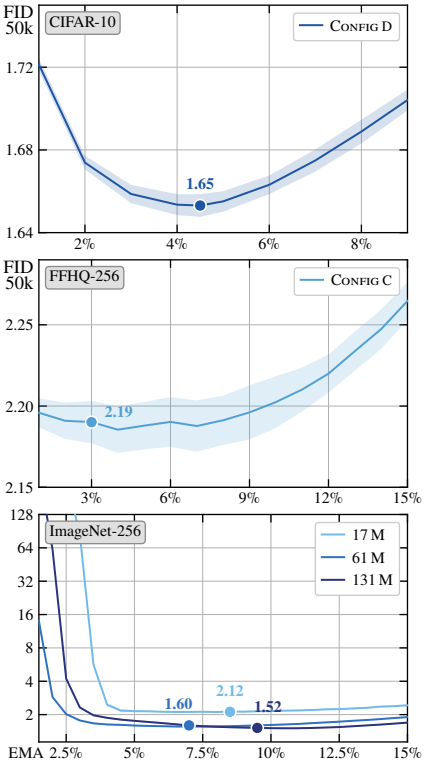


Figure 9: FID vs. post-hoc EMA length σ_{rel}. Mean of five runs; shading is standard error. Markers: released model σ_{rel} and five-run FID; not curve's minimum. We chose σ_{rel} on separate seeds before these runs, since picking the minimum on the reported seeds would bias FID. On FFHQ, the two differ by less than one standard error. ImageNet variance is smaller than the line used to draw the mean.

Efficiency implications for future generators. Computationally efficient adversarial generators imply promising future models using alternative autoencoders or generating directly in pixel space. Our results illustrate this opportunity: On FFHQ, direct pixel-space synthesis already realizes a sevenfold reduction in generation FLOPs relative to StyleGAN-XL [24]. On ImageNet, excluding decoding, R3GAN2-M requires approximately 0.05 TFLOPs per image, compared with 0.35 for iMF-XL/2 [56] and 0.24 for FACM-XL/2 [57]. The total cost is dominated by decoding; this finding motivates carrying efficient adversarial architectures into future generation systems with cheaper decoders (or none at all).

Untested scaling axes. While our approach is efficient and shows evidence of scaling across our S, M, L models, behavior at higher resolutions (512×512 and beyond), substantially larger models (≥ 500 M parameters), broader datasets, or under text-to-image conditioning remains untested. The self-attention diagnostic in Sec. 4.1, which we present as opening a path to multimodal extensions, is itself stable but slightly underperforms our pure-convolutional Config D. The magnitude-preserving recipe itself could plateau or require revisiting at larger scales, and further tests of attention-based layers likely require a larger, more diverse training distribution.

Broader impacts. Generative models are often computationally expensive, using significant energy, and may drive the development of new data centers and power plants. On the positive side, our generator needs less inference compute than prior models of similar quality (Tab. F.2), with potential influence on the development of other magnitude-preserving approaches. On the negative side, generative models are contentious as methods to create art, leading to the erasure of human-developed styles and low-quality mass-produced content, and our work makes it faster and cheaper to make such outputs.

Conclusion. We identify discriminator magnitude growth as a key bottleneck for scaling GAN training to diverse high-resolution data. By replacing scale-exploitable components with magnitude-preserving layers, we effectively eliminate this pathology, enable depth scaling, and obtain highly efficient one-step ImageNet 256×256 generators. These results suggest that the limits of modern GANs are not inherent to adversarial training, but depend strongly on controlling discriminator signal propagation.

Acknowledgments and Disclosure of Funding

NSF CAREER-2144956, Google's TPU Research Cloud, and Brown's secret NVIDIA B200 queue.

References

[1] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[2] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144, 2020.

[4] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016.

[5] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research (TMLR)*, 2022.

[6] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37:56424–56445, 2024.

[7] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[9] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.

[10] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24174–24184, 2024.

[11] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

[12] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025.

[13] Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026.

[14] Nick Huang, Aaron Gokaslan, Volodymyr Kuleshov, and James Tompkin. The GAN is dead; long live the GAN! A modern GAN baseline. *Advances in Neural Information Processing Systems*, 37:44177–44215, 2024.

[15] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard GAN. In *International Conference on Learning Representations (ICLR)*, 2019.

[16] Ruoyu Sun, Tiantian Fang, and Alexander Schwing. Towards a better global loss landscape of GANs. *Advances in Neural Information Processing Systems*, 33:10186–10198, 2020.

[17] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018.

[18] Alex Krizhevsky and Geoffrey E Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[20] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

[21] Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, et al. mHC: Manifold-constrained hyper-connections. In *International Conference on Machine Learning (ICML)*, 2026.

[22] Jeremy Bernstein. Modular manifolds. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250926. URL https://thinkingmachines.ai/blog/modular-manifolds/.

[23] Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected GANs converge faster. *Advances in Neural Information Processing Systems*, 34:17480–17492, 2021.

[24] Axel Sauer, Katja Schwarz, and Andreas Geiger. StyleGAN-XL: Scaling StyleGAN to large diverse datasets. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–10, 2022.

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

[26] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[27] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[28] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[29] Seyedmorteza Sadat, Jakob Buhmann, Derek Bradley, Otmar Hilliges, and Romann M Weber. LiteVAE: Lightweight and efficient variational autoencoders for latent diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

[30] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. In *International Conference on Learning Representations (ICLR)*, 2017.

[31] Andrew Brock, Soham De, Samuel L Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. In *International Conference on Machine Learning*, pages 1059–1071. PMLR, 2021.

[32] Andrew Brock, Soham De, and Samuel L Smith. Characterizing signal propagation to close the performance gap in unnormalized ResNets. In *International Conference on Learning Representations (ICLR)*, 2021.

[33] Devansh Arpit, Yingbo Zhou, Bhargava Kota, and Venu Govindaraju. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. In *International Conference on Machine Learning*, pages 1168–1176. PMLR, 2016.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pages 630–645. Springer, 2016.

[36] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

[37] Black Forest Labs. FLUX.2: Frontier visual intelligence. https://bfl.ai/blog/flux-2, 2025.

[38] Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. In *International Conference on Learning Representations (ICLR)*, 2025.

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

[40] Ruili Feng, Deli Zhao, and Zheng-Jun Zha. Understanding noise injection in GANs. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3284–3293. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/feng21g.html.

[41] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in neural information processing systems*, 33: 12104–12114, 2020.

[42] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[43] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[44] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10124–10134, 2023.

[45] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, 2022.

[46] 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. *Advances in neural information processing systems*, 30, 2017.

[47] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[48] George Stein, Jesse C. Cresswell, Rasa Hosseinzadeh, Yi Sui, Brendan Leigh Ross, Valentin Villecroze, Zhaoyan Liu, Anthony L. Caterini, J. Eric T. Taylor, and Gabriel Loaiza-Ganem. Exposing flaws of generative model evaluation metrics and their unfair treatment of diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[49] Tuomas Kynkäänniemi, Tero Karras, Miika Aittala, Timo Aila, and Jaakko Lehtinen. The role of ImageNet classes in Fréchet Inception distance. In *International Conference on Learning Representations (ICLR)*, 2023.

[50] Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Randomized autoregressive visual generation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 18431–18441, 2025.

[51] Funing Fu, Tenghui Wang, Guanyu Zhou, Junyong Cen, and Qichao Zhu. Equivalent flows, unequal learning: Clean-latent prediction in transformers. *arXiv preprint arXiv:2605.27102v3*, 2026.

[52] Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. SiT: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision (ECCV)*, 2024.

[53] 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 *International Conference on Learning Representations (ICLR)*, 2025.

[54] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *International Conference on Learning Representations (ICLR)*, 2026.

[55] Huijie Zhang, Aliaksandr Siarohin, Willi Menapace, Michael Vasilkovsky, Sergey Tulyakov, Qing Qu, and Ivan Skorokhodov. AlphaFlow: Understanding and improving MeanFlow models. In *International Conference on Learning Representations (ICLR)*, 2026.

[56] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[57] Yansong Peng, Kai Zhu, Yu Liu, Pingyu Wu, Hebei Li, Xiaoyan Sun, and Feng Wu. FACM: Flow-anchored consistency models. In *International Conference on Learning Representations (ICLR)*, 2026.

[58] Yuhta Takida, Masaaki Imaizumi, Takashi Shibuya, Chieh-Hsin Lai, Toshimitsu Uesaka, Naoki Murata, and Yuki Mitsufuji. SAN: Inducing metrizability of GAN with discriminative normalized linear layer. In *International Conference on Learning Representations (ICLR)*, 2024.

[59] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019.

[60] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34: 852–863, 2021.

[61] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018.

[62] Sangeek Hyun, MinKyu Lee, and Jae-Pil Heo. Scalable GANs with transformers. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. arXiv:2509.24935.

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

[64] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[65] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4917–4926. IEEE, 2019.

[66] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. Rethinking pre-training and self-training. *Advances in neural information processing systems*, 33:3833–3845, 2020.

[67] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

[68] Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *International Conference on Learning Representations (ICLR)*, 2025.

[69] Nicolas Boumal. *An Introduction to Optimization on Smooth Manifolds*. Cambridge University Press, 2023.

[70] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022.

[71] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

[72] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[73] Jean-Yves Franceschi, Mike Gartrell, Ludovic Dos Santos, Thibaut Issenhuth, Emmanuel de Bézenac, Mickaël Chen, and Alain Rakotomamonjy. Unifying GANs and score-based diffusion as generative particle models. *Advances in Neural Information Processing Systems*, 36:59729–59760, 2023.

[74] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

[75] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[76] Lucas Nestler. AVnorm. https://convergentthinking.sh/posts/avnorm/, 2026. Blog post.

[77] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

[78] Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=F7aAhfitX6.

[79] Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. In *International Conference on Learning Representations (ICLR)*, 2025.

[80] Gauthier Gidel, Reyhane Askari Hemmat, Mohammad Pezeshki, Rémi Le Priol, Gabriel Huang, Simon Lacoste-Julien, and Ioannis Mitliagkas. Negative momentum for improved game dynamics. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1802–1811. PMLR, 2019.

[81] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

[82] Linqi Zhou, Stefano Ermon, and Jiaming Song. Inductive moment matching. In *International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, 2025.

[83] Michael Tschannen, André Susano Pinto, and Alexander Kolesnikov. JetFormer: An autoregressive generative model of raw images and text. In *International Conference on Learning Representations (ICLR)*, 2025.

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

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

[86] Jiatao Gu, Tianrong Chen, David Berthelot, Huangjie Zheng, Yuyang Wang, Ruixiang Zhang, Laurent Dinh, Miguel Angel Bautista, Josh Susskind, and Shuangfei Zhai. STARFlow: Scaling latent normalizing flows for high-resolution image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025.

[87] 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 preprint arXiv:2406.06525*, 2024.

[88] 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 *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15781–15791, 2025.

[89] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning (ICML)*, 2023. arXiv:2301.11093.

[90] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[91] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15703–15712, 2025.

[92] Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. DDT: Decoupled diffusion transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[93] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *The Twelfth International Conference on Learning Representations*, 2024.

[94] Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations (ICLR)*, 2025.

[95] ZiDong Wang, Yiyuan Zhang, Xiaoyu Yue, Xiangyu Yue, Yangguang Li, Wanli Ouyang, and Lei Bai. Transition models: Rethinking the generative learning objective. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[96] Hansheng Chen, Kai Zhang, Hao Tan, Leonidas Guibas, Gordon Wetzstein, and Sai Bi. π-Flow: Policy-based few-step generation via imitation distillation. In *International Conference on Learning Representations (ICLR)*, 2026.

[97] Minguk Kang, Joonghyuk Shin, and Jaesik Park. StudioGAN: a taxonomy and benchmark of GANs for image synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[98] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

[99] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[100] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. TransGAN: Two pure transformers can make one strong GAN, and that can scale up. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[101] Kwonjoon Lee, Huiwen Chang, Lu Jiang, Han Zhang, Zhuowen Tu, and Ce Liu. ViTGAN: Training GANs with vision transformers. In *International Conference on Learning Representations (ICLR)*, 2022.

[102] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion GANs. In *International Conference on Learning Representations (ICLR)*, 2022.

[103] Zhendong Wang, Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Diffusion-GAN: Training GANs with diffusion. In *The Eleventh International Conference on Learning Representations*, 2023.

[104] Hui Lu, Albert Ali Salah, and Ronald Poppe. Compensation sampling for improved convergence in diffusion models. In *European Conference on Computer Vision (ECCV)*, 2024.

[105] Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10619–10629, June 2022.

[106] Rajhans Singh, Ankita Shukla, and Pavan Turaga. Polynomial implicit neural representations for large diverse datasets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2041–2051, 2023.

[107] Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[108] David So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V Le. Searching for efficient transformers for language modeling. *Advances in neural information processing systems*, 34: 6010–6022, 2021.

[109] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/fa6f2a469cc4d61a92d96e74617c3d2a-Paper.pdf.

[110] Yongsheng Yu, Wei Xiong, Weili Nie, Yichen Sheng, Shiqiu Liu, and Jiebo Luo. PixelDiT: Pixel diffusion transformers for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14273–14282, June 2026.

[111] Zehong Ma, Longhui Wei, Shuai Wang, Shiliang Zhang, and Qi Tian. DeCo: Frequency-decoupled pixel diffusion for end-to-end image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 43600–43610, June 2026.

[112] Heewoo Jun, Rewon Child, Mark Chen, John Schulman, Aditya Ramesh, Alec Radford, and Ilya Sutskever. Distribution augmentation for generative modeling. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5006–5019. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/jun20a.html.

[113] Yoshitaka Koike, Takumi Nakagawa, Hiroki Waida, and Takafumi Kanamori. Noiseless Diffusion-GAN: Scaling-based data augmentation for generative models. *Neural Networks*, 197:108458, 2026. ISSN 0893-6080. doi: 10.1016/j.neunet.2025.108458. URL https://www.sciencedirect.com/science/article/pii/S0893608025013395.

[114] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[115] George Dasoulas, Kevin Scaman, and Aladin Virmaux. Lipschitz normalization for self-attention layers with application to graph neural networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2456–2466. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/dasoulas21a.html.

[116] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://www.sciencedirect.com/science/article/pii/S0925231223011864.

[117] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. CvT: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22–31, October 2021.

[118] Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi Linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

[119] Sean Lie. Cerebras architecture deep dive: First look inside the hardware/software co-design for deep learning. *IEEE Micro*, 43(3):18–30, 2023.

[120] Dennis Abts, Jonathan Ross, Jonathan Sparling, Mark Wong-VanHaren, et al. Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads. In *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, pages 145–158, 2020. doi: 10.1109/ISCA45697.2020.00023.

Appendix Overview

A **Activation Magnitude Dynamics.** Defines the activation-magnitude statistics used throughout the paper and compares them for G and D across the roadmap (App. A.1); constructs a kernel discriminator showing that the R_1 and R_2 penalties do not prevent unbounded discriminator activation magnitude (App. A.2); and measures the amplification of persistent discriminative directions (App. A.3).

B **Weight Constraints.** Discusses alternative weight constraints and shows how row-wise ℓ_2 normalization still permits spectral sharpening (App. B.2).

C **Roadmap in Detail.** Details the changes that define Config B and Config C: the preliminary improvements of Config B, the magnitude-preserving architecture of Config C (ablated in Tab. 2), and the resampling operators.

D **Diagnostics in Detail.** Details the diagnostic ablations: explicit discriminator gain, alternative weight constraints, and magnitude-preserving self-attention.

E **Training and Efficient Implementation.** Lists the hyperparameters and schedules of our final models (Tab. E.1) and describes our efficient implementation: an explicit VJP for the gradient penalty, mixed precision, custom CUDA kernels, and measured training throughput.

F **Additional Results.** The evaluation protocol (App. F.1), the full ImageNet comparison (Tab. F.2), a breakdown of inference cost (Fig. F.1), the CIFAR-10 and FFHQ comparisons (Tabs. F.3 and F.4), further metrics (Tab. F.1), and qualitative samples.

G **Negative Results and Future Directions.** Ideas that did not help, directions for future work, and why one-step convolutional generators may suit dataflow accelerators.

A Activation Magnitude Dynamics

This section provides a more detailed analysis of the activation-magnitude behavior discussed in Sec. 2. First, we define the magnitude statistics used throughout the paper and compare their evolution in the generator and discriminator. Then, we examine the discriminator-side phenomenon more closely through an exactly analyzable scale-degenerate construction and an empirical analysis of persistent discriminative responses.

A.1 Magnitude Measurements

Measurement. Following Karras *et al.* [10], we measure activation magnitude using the spatial RMS of individual feature maps. First, for an activation tensor $x \in \mathbb{R}^{\mathrm{N}\times\mathrm{C}\times\mathrm{H}\times\mathrm{W}}$, we compute

$$r_{n,c}(x) = \sqrt{\frac{1}{\mathrm{HW}} \sum_{h=1}^{\mathrm{H}} \sum_{w=1}^{\mathrm{W}} x_{nchw}^2}. \tag{8}$$

Then, we aggregate across channels using either the mean

$$\frac{1}{\mathrm{N}} \sum_{n=1}^{\mathrm{N}} \frac{1}{\mathrm{C}} \sum_{c=1}^{\mathrm{C}} r_{n,c}(x), \tag{9}$$

or the maximum

$$\frac{1}{\mathrm{N}} \sum_{n=1}^{\mathrm{N}} \max_{c} r_{n,c}(x). \tag{10}$$

At each nonterminal resolution stage, we measure the signal after the residual blocks and immediately before the following resampling transition. The discriminator is measured on real images, the generator on generated ones. When the transition applies an accumulated-variance gain, we apply the same gain to the measured signal. At the final resolution, we measure the corresponding signal before the output head.

Discriminator magnitude dynamics. Figure A.1 shows both statistics throughout training for the discriminator stages of Configs A–D. The maximum and mean channel RMS exhibit the same qualitative behavior. In Configs A–B, activation magnitudes grow continuously through training, with the strongest growth concentrated in the later discriminator stages. This behavior becomes substantially

more severe in the deeper CONFIG B*, where peak magnitudes reach approximately 10^8. In contrast, after introducing the magnitude-preserving architecture in CONFIG C, discriminator magnitudes remain bounded throughout training, and the same remains true after depth scaling in CONFIG D. The mean channel RMS is typically 2–8× smaller (median about 3×) than the maximum channel RMS, but follows the same overall trajectory, indicating that the observed growth is not confined to a single exceptional channel.

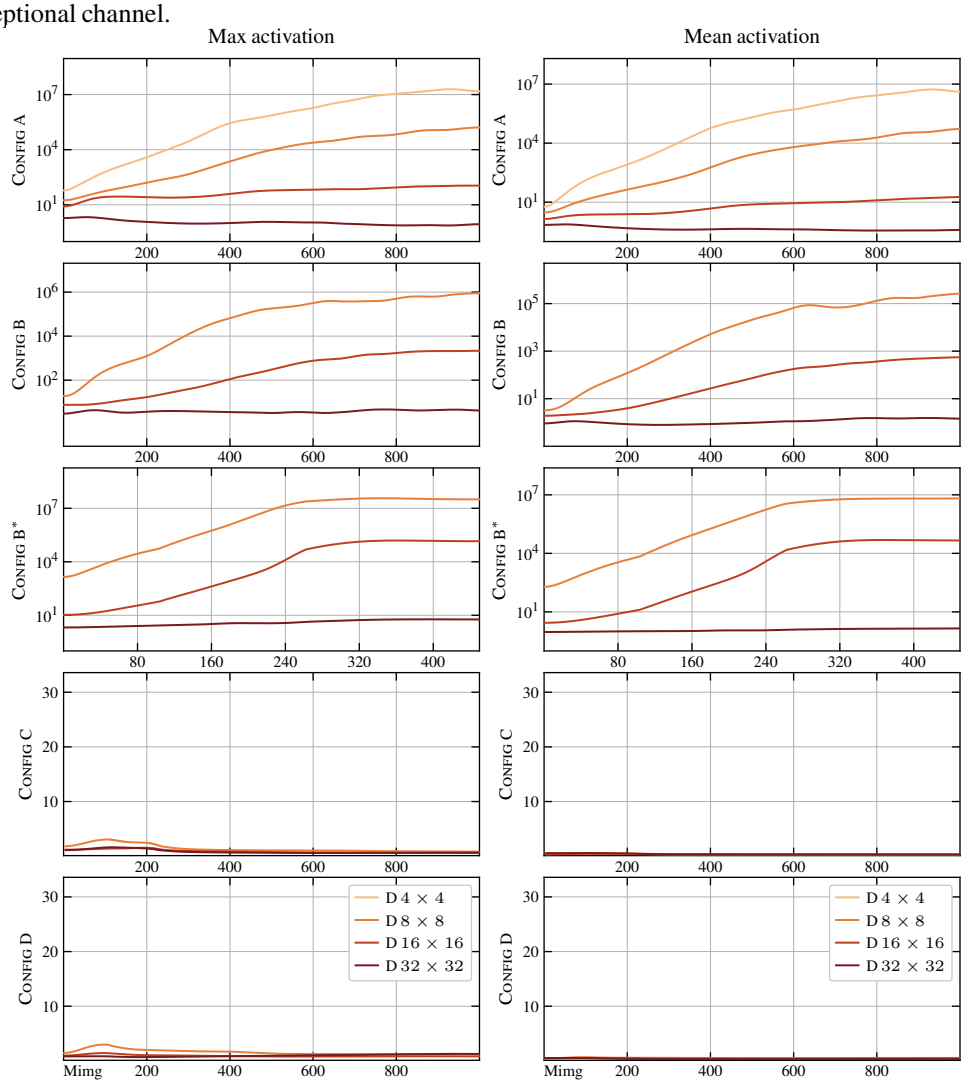


Figure A.1: Discriminator activation magnitudes throughout training, reported using the maximum (left) and mean (right) channel RMS defined in App. A.1.

Generator magnitude dynamics. Figure A.2 shows the corresponding generator measurements across the full ImageNet ablation roadmap. Mean channel RMS remains below about 1.3 and maximum channel RMS below about 4.5 throughout training in every configuration, including the pathological CONFIG B*. Thus, the generator has no magnitude pathology, yet magnitude-preserving parameterization still helps G in CONFIG C (Tab. 3), so it does more than correct runaway activation scale.

A.2 A Constructive Scale Degeneracy of the Empirical Discriminator

We now give an exactly analyzable example of the scale degree of freedom discussed in Sec. 2. Consider finite, disjoint sets of real and generated samples $\mathcal{X}_\mathcal{D}$ and $\mathcal{X}_\theta$. Define the scalar kernel score

$$\phi(x) = \max_{x_\mathcal{D} \in \mathcal{X}_\mathcal{D}} 2\exp\left(-\|x - x_\mathcal{D}\|_d^2\right) - 1, \tag{11}$$

where $\|\cdot\|_d$ denotes a distance and the kernel bandwidth is chosen such that $\phi(x_\theta) < 0$ for all $x_\theta \in \mathcal{X}_\theta$. Since $\phi(x_\mathcal{D}) = 1$ for every $x_\mathcal{D} \in \mathcal{X}_\mathcal{D}$ and both sets are finite, there exists a margin $m > 0$ such that

$$\phi(x_\mathcal{D}) \geq m, \qquad \phi(x_\theta) \leq -m. \tag{12}$$

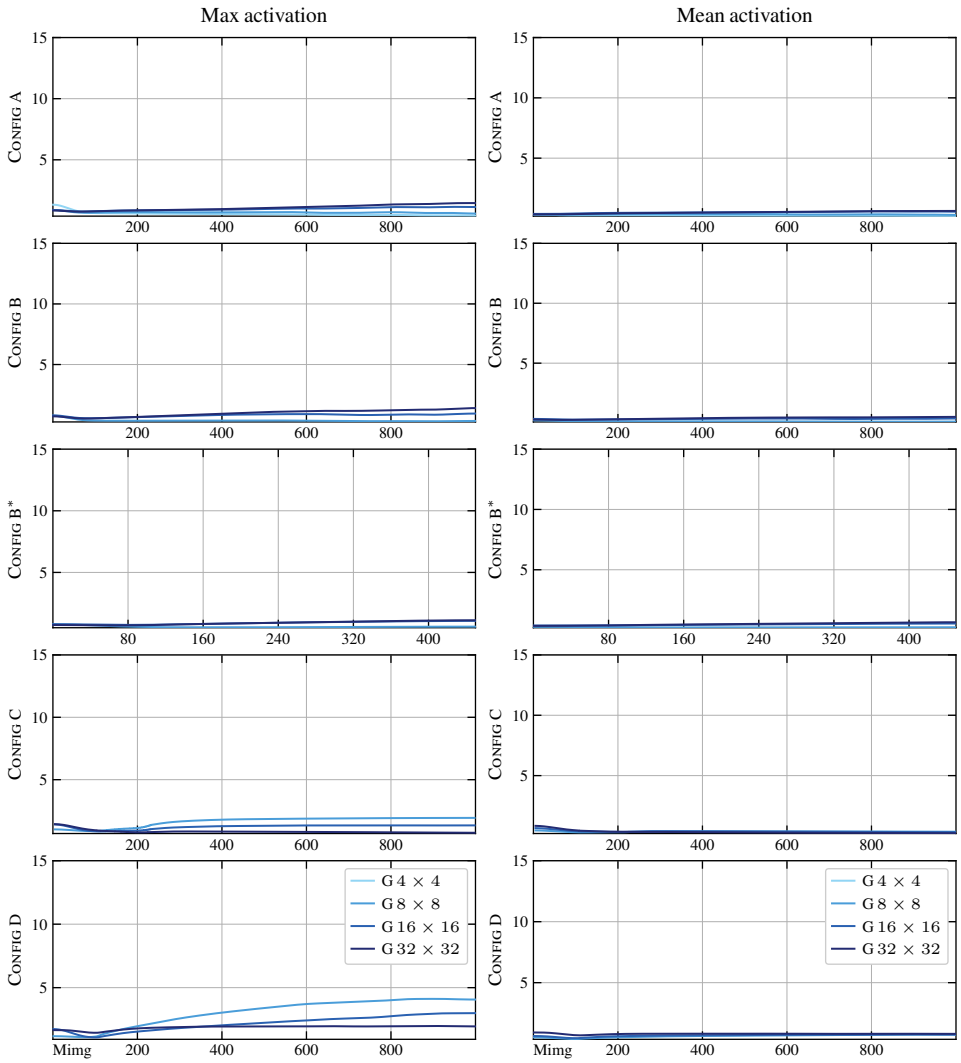


Figure A.2: Generator activation magnitudes throughout the ImageNet ablation roadmap, measured using maximum (left) and mean (right) channel RMS.

Thus, ϕ defines a fixed scalar representation that separates the two empirical sets.

Returning to the discriminator decomposition $D(x) = g(z(x))$ from Sec. 2, a sufficiently expressive discriminator can realize such a separator as one component of its representation $z(x)$ and map it to the scalar discriminator score through the downstream function g. Once this separating representation has been learned, the discriminator can change g while leaving the separator itself unchanged. To make this degree of freedom explicit, consider

$$D_\alpha(x) = g_\alpha(\phi(x)), \qquad g_\alpha(s) = e^{\alpha m} \tanh(\alpha s), \qquad \alpha > 0. \tag{13}$$

Across this family, $\phi(x)$ is fixed: its ordering, level sets, and separating boundary do not change. Only the downstream mapping g_α changes. The use of $\tanh$ is only for analytical convenience; the same saturating behavior can be represented by a piecewise-linear ReLU network.

Increasing α has four effects at once:

1. the empirical real–fake logit difference diverges to infinity;
2. consequently, the discriminator adversarial loss approaches zero;
3. the internal activation magnitude diverges;
4. the empirical R_1 and R_2 penalties approach zero.

In particular, this construction shows that empirical R_1 and R_2 do not in general prevent discriminator magnitude growth: both penalties can vanish along the same direction in which the internal activation magnitude diverges. We establish these four properties below.

Diverging logit separation and vanishing adversarial loss. For any $x_{\mathcal{D}} \in \mathcal{X}_{\mathcal{D}}$ and $x_\theta \in \mathcal{X}_\theta$, define the real–fake logit difference

$$M_\alpha(x_{\mathcal{D}}, x_\theta) = D_\alpha(x_{\mathcal{D}}) - D_\alpha(x_\theta). \tag{14}$$

Using Eq. 12 and the monotonicity of $\tanh$,

$$M_\alpha(x_{\mathcal{D}}, x_\theta) \geq 2e^{\alpha m} \tanh(\alpha m) \tag{15}$$

$$\longrightarrow \infty. \tag{16}$$

Thus, the empirical discriminator separates the same real and generated samples by an arbitrarily large score margin even though $\phi(x)$ itself is unchanged.

Writing the corresponding discriminator adversarial loss in its minimization form,

$$\mathcal{L}_{\text{adv}}(\alpha) = \frac{1}{|\mathcal{X}_{\mathcal{D}}||\mathcal{X}_\theta|} \sum_{x_{\mathcal{D}} \in \mathcal{X}_{\mathcal{D}}} \sum_{x_\theta \in \mathcal{X}_\theta} \text{softplus}(-M_\alpha(x_{\mathcal{D}}, x_\theta)), \tag{17}$$

we immediately obtain

$$\mathcal{L}_{\text{adv}}(\alpha) \longrightarrow 0. \tag{18}$$

Equivalently, under the maximization convention of the RpGAN objective in Eq. 1, the discriminator objective approaches its supremum. The improvement is entirely due to the changing downstream map g_α: the separating representation ϕ has not changed.

Diverging internal magnitude. The input to the saturating nonlinearity in Eq. 13 is

$$\ell_\alpha(x) = \alpha\phi(x). \tag{19}$$

For every empirical real or generated sample, Eq. 12 gives

$$|\ell_\alpha(x)| = \alpha|\phi(x)| \geq \alpha m \longrightarrow \infty. \tag{20}$$

Therefore, the discriminator can develop unbounded internal activation magnitude while its underlying one-dimensional representation remains fixed.

Vanishing R_1 and R_2. Because the empirical sets are finite and ϕ is differentiable at their samples, define

$$B = \max_{x \in \mathcal{X}_{\mathcal{D}} \cup \mathcal{X}_\theta} \|\nabla_x \phi(x)\|_2 < \infty. \tag{21}$$

Differentiating Eq. 13 yields

$$\nabla_x D_\alpha(x) = e^{\alpha m} \alpha \,\text{sech}^2(\alpha\phi(x))\, \nabla_x \phi(x). \tag{22}$$

Since $|\phi(x)| \geq m$ on all empirical samples and $\text{sech}^2(t) \leq 4e^{-2|t|}$,

$$\|\nabla_x D_\alpha(x)\|_2 \leq 4B\alpha e^{\alpha m} e^{-2\alpha m} \tag{23}$$

$$= 4B\alpha e^{-\alpha m} \longrightarrow 0. \tag{24}$$

Consequently, the empirical gradient penalties satisfy

$$R_1(\alpha) \longrightarrow 0, \qquad R_2(\alpha) \longrightarrow 0. \tag{25}$$

Taken together, these results exhibit an unbounded scale direction of the regularized empirical discriminator problem. By modifying only g_α, the discriminator can simultaneously increase empirical real–fake separation, reduce its adversarial loss, and reduce both gradient penalties, while the internal activation magnitude diverges and the separating representation $z(x) = \phi(x)$ remains fixed. Therefore, empirical R_1/R_2 regularization alone does not guarantee bounded discriminator activation magnitude.

Table A.1: **Magnitude preservation stops persistent directions from amplifying.** In CONFIG A their amplitude grows with depth, by thousands of times at 4×4; in the magnitude-preserving CONFIG C it stays at or below its starting level. Amplitude gains across discriminator stages, normalized to the reference checkpoint at approximately 200 M training images on ImageNet 256×256. Rows use the nearest available checkpoints to the indicated training progress. CONFIG C has no 4×4 residual blocks.

Images	32×32	16×16	8×8	4×4
200 M	1.00×	1.00×	1.00×	1.00×
400 M	0.93×	2.09×	12.70×	69.44×
600 M	0.92×	4.04×	241.67×	2473.03×
800 M	0.81×	5.13×	549.40×	7547.81×
1000 M	0.83×	8.06×	1581.97×	5144.91×

(a) CONFIG A

Images	32×32	16×16	8×8	4×4
200 M	1.00×	1.00×	1.00×	–
400 M	0.67×	0.67×	0.67×	–
600 M	0.80×	0.71×	0.69×	–
800 M	0.87×	0.72×	0.68×	–
1000 M	0.87×	0.73×	0.70×	–

(b) CONFIG C

A.3 Amplification of Persistent Discriminative Directions

The decomposition $D(x) = g(z(x))$ in Sec. 2 separates what the discriminator represents about an input from how strongly those represented differences affect its score. The construction in App. A.2 gives an exact limiting case in which a fixed discriminative representation is increasingly amplified through g. We now test whether amplification of persistent discriminative responses is visible in the ImageNet discriminators from our ablation roadmap, focusing on CONFIG A and CONFIG C. In a real network, $z(x)$ is high-dimensional and its coordinates can change throughout training; therefore, we seek discriminative response directions that persist functionally across checkpoints. These persistent directions provide an empirical proxy for class-dependent real–fake structure in $z(x)$ over training.

For each ImageNet class and discriminator stage, we compute the difference between the mean feature representation of real and generated images, and remove the mean difference across classes to isolate class-dependent real–fake variation. At a reference checkpoint, we identify the eight directions that explain the largest part of this variation. Since hidden feature coordinates may rotate, rescale, or mix throughout training, we match corresponding directions at later checkpoints through their response patterns across the 1000 ImageNet classes using Procrustes analysis. Subspace discovery and measurement use disjoint image sets.

CONFIG A. Table A.1a reports the amplitude of the persistent component at each discriminator stage, normalized to the reference checkpoint at approximately 200 M training images. The amplification is strongly depth-dependent. The 32×32 component remains near its reference magnitude, while the 16×16 component grows by roughly $8\times$. In the later stages, the growth becomes much more severe: by the end of training, the persistent component grows by a factor of about 1.6×10^3 at 8×8 and 5.1×10^3 at 4×4, precisely where the overall activation-magnitude pathology is strongest.

Focusing on the 4×4 stage, we compare these eight persistent directions against random directions and a control in which class identities are randomly permuted before comparing response patterns. They carry a disproportionately large share of the total squared class-dependent real–fake variation: a median of 17.6%, compared with approximately 0.8% for either control, or roughly $22\times$ as much as unrelated subspaces of the same dimension.

CONFIG C. We repeat the same analysis for the magnitude-preserving CONFIG C. While persistent discriminative directions remain identifiable, as shown in Tab. A.1b, the amplification observed in CONFIG A disappears. Across all three discriminator stages, their amplitudes remain close to or below the reference magnitude throughout training.

These persistent directions also capture only modestly more variation than the control subspaces. At 32×32, they capture a median of 1.46% of the squared class-dependent real–fake variation, against 0.91% for random directions and 0.75% after permuting class identities; at 16×16, 1.26% against 0.78% (random) and 0.88% (permuted); at 8×8, 1.35% against 0.75% and 0.79%.

Together, these results show that magnitude pathology coincides with strong amplification of persistent discriminative directions in CONFIG A, while both are absent in CONFIG C.

B Weight Constraints

Section 3 adopts row-wise ℓ_2 weight normalization as the magnitude-preserving parameterization for linear layers. This choice corresponds to preserving activation variance under the approximately i.i.d. input assumption used throughout our magnitude-propagation analysis. Here we contrast this statistical criterion with stronger constraints that make progressively fewer assumptions about the input, and then examine how correlated activations can still produce amplification under row-wise ℓ_2 normalization.

B.1 Alternative Weight Constraints

ℓ_1 normalization. A stronger row-wise constraint can be obtained by retaining only the assumption that each input coordinate has variance σ_x^2, while allowing arbitrary correlations between coordinates. Since $|\operatorname{Cov}(x_j, x_k)| \le \sigma_x^2$,

$$\operatorname{Var}[\mathbf{w}^\top \mathbf{x}] = \sum_{j,k} w_j w_k \operatorname{Cov}(x_j, x_k) \tag{26}$$

$$\le \sigma_x^2 \sum_{j,k} |w_j||w_k| = \sigma_x^2 \|\mathbf{w}\|_1^2. \tag{27}$$

Thus, $\|\mathbf{w}\|_1 = 1$ prevents variance amplification even under worst-case channel correlation. Under approximately independent inputs, however, this constraint is generally variance contractive because $\|\mathbf{w}\|_2 \le \|\mathbf{w}\|_1 = 1$. Related ℓ_1 constraints appear, often together with non-negativity, in stochastic matrices and softmax-like mappings [67], and have recently been adopted in hyper-connections [21, 68]. More generally, row-wise ℓ_p normalization with $1 < p < 2$ may offer an intermediate way to limit spectral sharpening without the stronger contraction of ℓ_1 normalization; we leave this direction to future work.

Singular-value constraints. A different approach is to constrain the worst-case amplification of the full linear transformation rather than each output coordinate separately. Spectral normalization [42] constrains the largest singular value of $\mathbf{W}$, typically to $\sigma_{\max}(\mathbf{W}) = 1$, which guarantees $\|\mathbf{W}\mathbf{x}\|_2 \le \|\mathbf{x}\|_2$ for every input $\mathbf{x}$. Its implication for per-coordinate activation magnitude depends on the input and output dimensionalities; for a square weight matrix, spectral normalization is generally variance contractive on average across coordinates. Another related technique is orthonormalization [69], which makes the weight matrix (semi-)orthogonal, so all singular values are 1. This is a stronger constraint than spectral normalization, since it fixes all singular values rather than only the largest.

B.2 Spectral Sharpening Under Row-Wise ℓ_2 Normalization

The variance-preservation argument for row-wise ℓ_2 normalization assumes approximately independent, or more generally isotropic, inputs. However, a normalized linear layer can itself introduce correlations between activation coordinates, which can then be amplified by subsequent normalized layers. Consider

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_2), \tag{28}$$

and two linear layers

$$\mathbf{y} = \mathbf{W}_1 \mathbf{x}, \qquad z = \mathbf{W}_2 \mathbf{y}, \tag{29}$$

with

$$\mathbf{W}_1 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \qquad \mathbf{W}_2 = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}. \tag{30}$$

Every row of $\mathbf{W}_1$ and $\mathbf{W}_2$ has unit ℓ_2 norm. Thus, the first layer preserves the variance of each individual output coordinate,

$$\mathbf{y} = \begin{bmatrix} x_1 \\ x_1 \end{bmatrix}, \qquad \operatorname{Var}(y_1) = \operatorname{Var}(y_2) = 1, \tag{31}$$

but makes the two coordinates perfectly correlated. The second layer then adds two aligned copies of the same feature:

$$z = \frac{1}{\sqrt{2}} x_1 + \frac{1}{\sqrt{2}} x_1 = \sqrt{2} x_1, \qquad \operatorname{Var}(z) = 2. \tag{32}$$

Thus, two individually row-wise ℓ_2-normalized layers can still amplify activation variance through correlations introduced between them.

Singular-value interpretation. The same mechanism generalizes to higher dimensions. Let $\mathbf{W} \in \mathbb{R}^{m \times d}$ be a row-wise ℓ_2-normalized layer, with rows $\mathbf{w}_i$ satisfying $\|\mathbf{w}_i\|_2 = 1$. Then

$$\|\mathbf{W}\|_F^2 = \sum_{i=1}^{m} \|\mathbf{w}_i\|_2^2 = m. \tag{33}$$

Let $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be its singular value decomposition, with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$. Thus, row-wise normalization fixes the total squared singular value mass,

$$\sum_i \sigma_i^2 = \|\mathbf{W}\|_F^2 = m, \tag{34}$$

but does not constrain how this mass is distributed across singular directions. Under spectral sharpening, a larger fraction of this fixed mass becomes concentrated in a small number of dominant singular values.

For an isotropic input $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, the output $\mathbf{y} = \mathbf{W}\mathbf{x}$ has covariance

$$\mathrm{Cov}(\mathbf{y}) = \mathbf{W}\mathbf{W}^\top = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^\top\mathbf{U}^\top, \tag{35}$$

whose nonzero eigenvalues are σ_i^2. Therefore, a subsequent row-wise normalized layer can amplify a dominant correlated direction. In particular, if $\mathbf{u}_1$ is the leading left singular vector and the next layer contains the unit-norm row $\mathbf{a}^\top = \mathbf{u}_1^\top$, then

$$\mathrm{Var}[\mathbf{a}^\top\mathbf{y}] = \mathbf{u}_1^\top \mathrm{Cov}(\mathbf{y})\mathbf{u}_1 = \sigma_1^2. \tag{36}$$

Thus, even though every row of $\mathbf{W}$ has unit norm, spectral sharpening can create correlated directions whose variance exceeds one and can be amplified by subsequent row-wise normalized layers.

Practical implications. Spectral sharpening concentrates a fixed weight-magnitude budget into fewer dominant singular directions, reducing the effective rank of the transformation and potentially carrying a representational cost. In our ordinary row-wise ℓ_2-normalized runs, this effect does not become severe enough to produce pathological magnitude growth. It becomes more pronounced in diagnostic settings where amplification is strongly favored; the explicit discriminator-gain ablation in App. D.1 provides one such example, where spectral sharpening allows amplification to re-emerge inside the discriminator trunk.

C Roadmap in Detail

This appendix provides additional implementation details for the architectural changes summarized in Sec. 4. First, we describe the preliminary improvements that define CONFIG B, followed by the magnitude-preserving architecture and cumulative component ablations that define CONFIG C. Then, we give a detailed specification of the shuffle-based resampling operator introduced in CONFIG B.

C.1 Preliminary Improvements

In the main paper, we incorporate a collection of preliminary architectural and optimization changes on top of the baseline.

Architecture and optimization. First, we increase the group size of the 3×3 convolution inside each residual block from 16 to 32; R3GAN's residual blocks follow ConvNeXt [70] but use grouped rather than depthwise convolutions. This yields a modest FID improvement with negligible impact on parameter count, while improving GPU efficiency. We further make two changes to the baseline architecture and training procedure. Following EDM, which also removes its lowest-resolution layers [9], we remove the 4×4 residual blocks (the 4×4 bases remain); in our setting, we find they contribute disproportionately to overfitting. In addition, we replace the standard exponential moving average (EMA) [28, 39, 60] of generator weights with EDM2's power EMA [10], which also enables post-hoc tuning [10] of the EMA decay after training.

Bias parameterization. We also revisit bias parameterization, which is often treated as an implementation detail. EDM2 [10] removes explicit biases and instead appends a constant channel of ones at the network input; in our setting, this modification significantly degrades both stability and sample quality. R3GAN [14] uses biases only within residual blocks, after the first 1×1 convolution and the middle 3×3 convolution. We find that removing the post-3×3 bias is safe while retaining the remaining bias term. Moreover, the remaining bias can be absorbed into the preceding 1×1 convolution by treating it as an additional constant input channel within each residual block.

Resolution transitions. We identify the resolution transition layers as an avoidable source of information loss. In R3GAN [14], an upsampling transition from $\mathrm{H} \times \mathrm{W} \times 2\mathrm{C}$ to $2\mathrm{H} \times 2\mathrm{W} \times \mathrm{C}$ is implemented as a channel-reducing 1×1 convolution followed by bilinear interpolation [39]: the projection irreversibly compresses the representation before spatial expansion, while interpolation only redistributes the remaining signal. To better preserve information through these transitions, we replace the learned projection-and-interpolation operation with non-parametric phase rearrangement in both G and D: pixel shuffle [71] with channel duplication for upsampling, and inverse shuffle with channel averaging for downsampling, as in [38]. We additionally apply a lightweight binomial low-pass filter [39] after upsampling and before downsampling to suppress grid-phase artifacts introduced by the rearrangement. We describe the unified resampling construction and its Nyquist properties in App. C.3.

Noise injection. Unlike on the smaller and more homogeneous datasets considered in the original R3GAN study, we find that noise injection [39] is beneficial on ImageNet 256×256. We reintroduce noise injection by concatenating a standard Gaussian noise channel to the input of each generator residual block. One interpretation uses the generator's neural tangent kernel (NTK) [72]: updates induced by the discriminator are filtered through it [73], and a low effective-rank NTK couples many samples to move in a few shared directions, which on a diverse benchmark such as ImageNet 256×256 limits transport and drops modes. Injected noise adds sample-specific degrees of freedom that diversify tangent features, reducing off-diagonal coupling and raising effective rank [40], so samples move more independently and cover more of the data support.

C.2 Magnitude-Preserving Architecture

Starting from CONFIG B, we apply the magnitude-preserving parameterization developed in Sec. 3 to both G and D. The main paper ablates its application to the two networks (Tab. 3) and the cumulative effect of its components (Tab. 2); here, we describe the full CONFIG C architecture, one paragraph per row of Tab. 2.

Weight normalization. We apply row-wise ℓ_2 weight normalization (Eq. 3) [10, 74] to all learnable linear operators, including fully connected layers, convolutions, embeddings, and the learned 4×4 bases. We additionally use a learnable gain on the final layer of G, as in EDM2 [10], since the target distribution in latent space does not necessarily have unit variance. We do not use a corresponding output gain in D; App. D.1 examines this parameterization separately. Introducing row-wise ℓ_2 weight normalization reduces FID from 19.20 to 9.82.

Magnitude-preserving LReLU and weight centering. Although weight centering (Eq. 5) [32] is only required for the linear layer preceding LReLU in the mean-field [30] analysis, we find it empirically beneficial to also center the linear layer following LReLU, and therefore apply weight centering to all linear layers adjacent to LReLU. For the nonlinearity itself, we rescale LReLU by the inverse of its expected standard-deviation factor [33] given by Eq. 4, making the activation approximately magnitude preserving under the same Gaussian assumption. These changes further reduce FID from 9.82 to 9.71.

Magnitude-preserving residuals. Residual connections [34] are handled via the parameterization [32] in Eq. 6. We implement α_i as a channel-wise vector initialized to zero, so that each block starts in an exact identity regime consistent with pre-activation ResNet [35] behavior. While Eq. 6 permits the limit $\alpha_i \to \infty$, which effectively makes the network non-residual, we do not observe benefits from allowing the residual branch to dominate the skip connection. Therefore, we constrain $|\alpha_i| \in [0, 1)$ through the reparameterization $\alpha_i = \tanh(\alpha_i')$, where α_i' is unconstrained. Although β_i stabilizes the variance seen by the residual branch [32], the variance on the main path still accumulates across blocks as a consequence of preserving the identity mapping. Following prior work [31, 32], we reset the running scale at the end of each resolution stage by applying the final accumulated β in the transition layer between stages. Magnitude-preserving residuals further improve FID from 9.71 to 9.29.

Network heads. As observed in CONFIGS A–B*, discriminator magnitude growth is most pronounced in the late blocks immediately preceding the classifier. Therefore, we replace the original linear classifier head [14] with a small feed-forward network that mirrors a residual-block-style computation: a 1×1 convolution, LReLU, a global grouped convolution, LReLU, and a final fully connected projection. We also replace the generator head with a symmetric feed-forward design. With these heads, FID improves from 9.29 to 8.07.

Forced weight normalization and optimization. Finally, ℓ_2 weight normalization induces growth in the unnormalized weights [10, 74], which can distort the effective learning rate unless explicitly controlled. We adopt forced weight normalization [10] as in EDM2, but modify its implementation

to suit GAN training: instead of normalizing raw weights on-the-fly during the forward pass, we materialize the normalization as a post-processing operation after each optimizer update. This avoids modifying the model parameters between the multiple discriminator evaluations performed within a training step. We also adopt the inverse-square-root learning rate schedule [10, 75] from EDM2, as constant learning rates no longer facilitate convergent training under forced normalization; App. E.1 defines the schedule. Together, forced weight normalization and the new learning-rate schedule reduce FID from 8.07 to 5.37.

C.3 Resampling Architecture

The resolution transitions in CONFIG B change spatial resolution by a factor of two while the channel ratio between the low- and high-resolution representations can be 1, 2, or 4. We implement all three cases with the same phase-based construction. Let

$$q = \frac{\mathrm{C}_{\mathrm{low}}}{\mathrm{C}_{\mathrm{high}}} \in \{1, 2, 4\}, \tag{37}$$

where $\mathrm{C}_{\mathrm{low}}$ is the channel count at the low spatial resolution and $\mathrm{C}_{\mathrm{high}}$ at the high one. A 2×2 high-resolution cell contains four spatial phases. For upsampling, each high-resolution output channel has q independent values available in the channel dimension; we duplicate these values to obtain four phases, apply pixel shuffle [71], and then low-pass filter the resulting spatial lattice. Downsampling applies the matched operations in reverse: low-pass filtering, pixel unshuffle, and averaging the redundant phases. Denoting the spatial filtering operation by $\mathcal{F}$,

$$\begin{aligned} \mathrm{Up}_q(x) &= \mathcal{F}\Big[\mathrm{shuffle}\Big(\mathrm{dup}_{4/q}(x)\Big)\Big], \\ \mathrm{Down}_q(x) &= \mathrm{avg}_{4/q}[\mathrm{unshuffle}(\mathcal{F}[x])]. \end{aligned} \tag{38}$$

Here $\mathrm{dup}_{4/q}$ expands each group of q independent phase values to four, while $\mathrm{avg}_{4/q}$ performs the corresponding reduction. Therefore, the three channel ratios differ only in how many of the four spatial phases are already represented independently:

q	Upsampling	Downsampling
1	duplicate $1 \to 4$	average $4 \to 1$
2	duplicate $2 \to 4$	average $4 \to 2$
4	no duplication	no averaging

Thus, $q = 1$ recovers conventional $2\times$ interpolation, $q = 4$ reduces to ordinary pixel shuffle/unshuffle with low-pass filtering, and $q = 2$ provides an intermediate case with two independently specified phases.

Two-phase case. The $q = 2$ case is the only one for which the phase arrangement is not unique. Let

$$\mathbf{P}_c = \begin{bmatrix} p_{00,c} & p_{01,c} \\ p_{10,c} & p_{11,c} \end{bmatrix} \tag{39}$$

denote the four spatial phases of high-resolution channel c, where $p_{ij,c}$ is the value at phase (i, j). For upsampling, the two corresponding low-resolution channels x_{2c} and x_{2c+1} are duplicated along one spatial axis. We alternate this direction across output channels:

$$\mathbf{P}_c = \begin{cases} \begin{bmatrix} x_{2c} & x_{2c+1} \\ x_{2c} & x_{2c+1} \end{bmatrix}, & c \equiv 0 \pmod 2, \\ \begin{bmatrix} x_{2c} & x_{2c} \\ x_{2c+1} & x_{2c+1} \end{bmatrix}, & c \equiv 1 \pmod 2. \end{cases} \tag{40}$$

Alternating the orientation prevents all channels from sharing the same horizontal or vertical duplication pattern. For downsampling, pixel unshuffle recovers $\mathbf{P}_c$, after which the corresponding phase pairs are averaged:

$$\begin{aligned} c \equiv 0 \pmod 2 &: \quad x_{2c} = \frac{p_{00,c} + p_{10,c}}{2}, \quad x_{2c+1} = \frac{p_{01,c} + p_{11,c}}{2}, \\ c \equiv 1 \pmod 2 &: \quad x_{2c} = \frac{p_{00,c} + p_{01,c}}{2}, \quad x_{2c+1} = \frac{p_{10,c} + p_{11,c}}{2}. \end{aligned} \tag{41}$$

The $q = 1$ and $q = 4$ cases follow directly from Eq. 38: the former duplicates one value into all four phases, while the latter already provides all four phases and requires no duplication.

Interpolation as the one-phase case. Although the $q = 1$ construction is expressed using phase duplication and pixel shuffle, it is equivalent to conventional center-aligned $2\times$ interpolation. In one dimension, duplicating each sample onto the two high-resolution phases can be written as zero insertion followed by a two-tap filter:

$$\mathrm{dup}_2(x) = (\uparrow_2 x) * [1, 1]. \tag{42}$$

Thus, applying a post-duplication filter h gives

$$\mathrm{dup}_2(x) * h = (\uparrow_2 x) * g, \qquad g = [1, 1] * h. \tag{43}$$

For a conventional center-aligned interpolation kernel g, the even and odd output phases both preserve constants,

$$\sum_k g[2k] = \sum_k g[2k + 1] = 1, \tag{44}$$

and hence its response at Nyquist satisfies

$$G(\pi) = \sum_k g[2k] - \sum_k g[2k + 1] = 0. \tag{45}$$

For finite-support filters, this Nyquist zero provides the $[1, 1]$ factor in Eq. 43. Therefore, the one-phase construction can realize the usual family of center-aligned $2\times$ interpolation filters through the choice of h, rather than being tied to a particular interpolation rule. The same argument applies separably in two dimensions.

In our models, we use the centered binomial filter

$$h = \frac{1}{4}[1, 2, 1]. \tag{46}$$

For $q = 1$, the corresponding effective interpolation kernel is

$$g = [1, 1] * h = \frac{1}{4}[1, 3, 3, 1], \tag{47}$$

which recovers center-aligned bilinear interpolation. The same filter is used for the $q = 2$ and $q = 4$ cases.

Nyquist-null filtering. For $q = 2$ and $q = 4$, neighboring pixels after shuffle may originate from independently specified channel phases, so disagreement between phases can produce alternating grid artifacts. The binomial filter in Eq. 46 removes the corresponding Nyquist modes exactly. Its centered one-dimensional frequency response is

$$H(\omega) = \frac{1}{4}\left(e^{-i\omega} + 2 + e^{i\omega}\right) = \cos^2(\omega/2), \tag{48}$$

and therefore

$$H(\pi) = 0. \tag{49}$$

For the separable two-dimensional filter,

$$H_{2\mathrm{D}}(\omega_x, \omega_y) = H(\omega_x)H(\omega_y), \tag{50}$$

so

$$H_{2\mathrm{D}}(\pi, 0) = H_{2\mathrm{D}}(0, \pi) = H_{2\mathrm{D}}(\pi, \pi) = 0. \tag{51}$$

Thus, the filter exactly annihilates horizontal and vertical one-pixel alternating modes, as well as the canonical 2×2 checkerboard mode $(-1)^{i+j}$. We apply it after phase placement for upsampling and before phase extraction for downsampling in all three channel-ratio cases.

D Diagnostics in Detail

The diagnostic ablations in Sec. 4.1 examine several alternative ways of controlling discriminator scale and extending the magnitude-preserving design. Each diagnostic modifies CONFIG D and is compared against it, trained as described in Sec. 4.

D.1 Explicit Discriminator Gain

Naive output gain. We examine in more detail the learnable discriminator output gain

$$D_{\psi,\lambda}(x) = \lambda D_\psi(x). \tag{52}$$

Naively introducing λ causes the gain to rapidly collapse while intermediate discriminator magnitudes increase sharply. This behavior is consistent with its interaction with the gradient penalties. Substituting $D_{\psi,\lambda}$ into Eq. 2 gives

$$\begin{aligned} R_1(\psi,\lambda) &= \frac{\gamma}{2}\lambda^2 \mathbb{E}_{x\sim p_{\mathcal{D}}}\left[\|\nabla_x D_\psi(x)\|^2\right], \\ R_2(\theta,\psi,\lambda) &= \frac{\gamma}{2}\lambda^2 \mathbb{E}_{x\sim p_\theta}\left[\|\nabla_x D_\psi(x)\|^2\right]. \end{aligned} \tag{53}$$

Thus, both penalties scale quadratically with λ, creating a direct incentive for the gain to decrease. As the output gain shrinks, the discriminator compensates through spectral sharpening, and the magnitude pathology re-emerges in the trunk.

Detached output gain. To remove the direct incentive to shrink λ, we stop the penalty gradients with respect to the gain while retaining its value in R_1 and R_2. With this modification, the behavior reverses: λ grows steadily throughout training, while intermediate discriminator activations contract about fivefold, mostly within the first few million images (Fig. 5). Thus, an explicit output gain can absorb the increase in discriminator scale without producing magnitude growth in the trunk when it is decoupled from the penalty gradients. However, this does not improve sample quality over CONFIG D (Tab. 4). A plausible explanation is that the effective regularization on D_ψ still scales as $\gamma\lambda^2$; as λ grows, the discriminator trunk is increasingly constrained by the gradient penalties.

D.2 Alternative Weight Constraints

We evaluate the stronger ℓ_1 and spectral weight constraints discussed in App. B.1 as alternatives to row-wise ℓ_2 normalization in the discriminator. Applying either constraint to all learnable layers in D causes immediate training failure, so in the reported experiments we restrict them to the discriminator residual blocks. Both variants also require a $5\times$ lower learning rate to avoid divergence.

Both alternatives perform substantially worse than CONFIG D. With row-wise ℓ_1 normalization, the generator produces primarily low-frequency, blurred color blobs and FID deteriorates severely. Spectral normalization produces recognizable samples but remains far behind CONFIG D in fidelity. Consistent with the variance contraction predicted in App. B.1, discriminator activation magnitudes decrease throughout training in both cases, accompanied by r_t approaching zero and very small discriminator input gradients (Tab. 4). These observations suggest that the discriminator becomes too contractive to provide an effective adversarial signal.

D.3 Self-Attention

Attention sharpness. For the attention experiment in Sec. 4.1, we insert a single attention block at the middle of the 16×16 stage in both G and D, using 4 heads with head dimension 64. To prevent attention sharpness from being controlled through the magnitudes of the query and key vectors, we normalize query and key tokens with row-wise ℓ_2 normalization, replacing dot-product attention with cosine attention [10] and constraining the unscaled logits to $[-1, 1]$. Since this dynamic range is too restrictive at 16×16 resolution, even the maximally favored token can receive at most 2.8% probability mass. Therefore, we introduce a per-head learnable sharpness a, initialized to 1, to expand the logit range to $[-a, a]$. In our experiments, a stabilizes at approximately 7.38, close to the standard dot-product attention scale in our parameterization, which corresponds to $a = 8$.

Magnitude-preserving aggregation. Let $\hat{\mathbf{Q}}$ and $\hat{\mathbf{K}}$ denote the row-normalized query and key matrices. The attention matrix is

$$\mathbf{A} = \mathrm{softmax}\left(a\hat{\mathbf{Q}}\hat{\mathbf{K}}^\top\right). \tag{54}$$

Row-wise softmax performs an ℓ_1 normalization, and the magnitude of the output $\mathbf{AV}$ depends on the concentration of each row of $\mathbf{A}$: sharper attention increases the expected magnitude of $\mathbf{AV}$ even when the value tokens are approximately i.i.d. This provides an implicit degree of freedom to modulate activation magnitude through attention peakiness. We remove this mechanism with AVNorm [76], which normalizes each output token by its RMS norm, making the output magnitude invariant to attention peakiness. We also apply row-wise RMS normalization to $\mathbf{V}$ [10] so that attention behavior is not confounded by variation in value magnitudes.

Attention sinks. Finally, softmax attention cannot represent "attend to nothing," which can manifest as attention sinks [77] and massive activations [78]. A direct remedy is to add a learnable sink that absorbs probability mass; however, pure output attenuation is nullified by AVNorm. Therefore, we instead append a learned, context-independent bias token to $\mathbf{K}$ and $\mathbf{V}$ [79], which can be attended to when no context token is informative. Under this design, we do not observe unintended attention sinks or massive activations.

E Training and Efficient Implementation

This appendix provides the training and implementation details for our final models, complementing the architectural specification in App. C. First, we describe the optimization schedules, hyperparameters, and augmentation settings used for training. Then, we discuss the main implementation optimizations used to reduce training cost, including an explicit differentiable VJP for discriminator gradient penalties, mixed-precision training, and specialized CUDA implementations for the dominant network operations. Finally, we benchmark the complete training step (D update with R_1, and G update, both with the optimizer) across our implementation stages and six GPU models.

E.1 Training Hyperparameters and Schedules

Optimization and gradient penalty. Following R3GAN [14], we optimize G and D with Adam using identical settings, with $\beta_1 = 0$ and $\epsilon = 10^{-8}$. We do not use momentum, as it can be detrimental to adversarial game dynamics [80]. The learning rate and β_2 are scheduled over training as described below. For gradient penalties, R_1 and R_2 essentially reduce to the same objective in the linearized regime close to equilibrium [17]. Their practical distinction is most relevant early in training, where R_2 provides an additional safeguard against the gradient explosion on generated samples observed in R3GAN. All roadmap configurations (CONFIGS A–D) use both penalties. For the final, fully trained models, we find this safeguard unnecessary with our magnitude-preserving design and therefore set $\gamma_{R_2} = 0$ to reduce computational cost. Empirically, this change does not degrade training or sample quality when γ_{R_1} is increased to maintain the overall gradient penalty strength.

Augmentation. We enable dataset x-flips in all experiments. For ImageNet, we create a mirrored copy of each RGB image before encoding both versions with the FLUX.2 encoder. We also use non-leaking augmentation [41], considering four categories: pixel blitting, geometric, color, and cutout augmentations. We apply all four categories to CIFAR-10 and FFHQ. Although color augmentations can be generalized to the latent space of the FLUX.2 autoencoder (App. G), we observe no benefit from doing so and therefore omit color augmentation for ImageNet. For each real–fake pair in the RpGAN loss (Eq. 1), we apply the same sampled augmentation transformations to both samples. The per-operation probability multipliers are specified for each model in Tab. E.1, while the overall augmentation probability is scheduled over training as described below.

Hyperparameter schedules. We schedule the learning rate η, the Adam parameter β_2, the R_1 regularization strength γ_{R_1}, and the augmentation probability p over training. Our schedules build on the burn-in strategy of R3GAN [14], replacing its cosine interpolation during burn-in with the schedules defined below. Following EDM2 [10], we additionally introduce linear learning-rate warmup before burn-in and inverse-square-root learning-rate decay afterward. Let n denote the number of training images processed, and let n_{w} denote the warmup endpoint and n_{b} the burn-in endpoint. The parameters β_2, γ_{R_1}, and p remain at their initial values during warmup and at their final values after burn-in. We express their evolution using the normalized progress

$$u(n) = \mathrm{clip}\left(\frac{n - n_{\mathrm{w}}}{n_{\mathrm{b}} - n_{\mathrm{w}}},\, 0,\, 1\right), \tag{55}$$

with subscripts 0 and 1 denoting initial and final values.

We interpolate p linearly to change the augmentation probability at a constant absolute rate during burn-in. For γ_{R_1}, we use log-linear interpolation to change the regularization strength at a constant relative rate:

$$p(n) = (1 - u(n))p_0 + u(n)p_1, \qquad \gamma_{R_1}(n) = \gamma_0 \left(\frac{\gamma_1}{\gamma_0}\right)^{u(n)}. \tag{56}$$

For Adam, equal increments in β_2 correspond to increasingly large changes in its averaging horizon as β_2 approaches one. Therefore, we interpolate the horizon log-linearly, so that the memory length of the

Table E.1: **Hyperparameters for each experiment.** ImageNet-S is the same as CONFIG D in Sec. 4 with tuned training parameters.

Hyperparameter	CIFAR-10	FFHQ	ImageNet		
			S	M	L
Data dimension	$32 \times 32 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$
Domain dimension	$32 \times 32 \times 3$	$256 \times 256 \times 3$	$32 \times 32 \times 32$	$32 \times 32 \times 32$	$32 \times 32 \times 32$
Domain	Pixel	Pixel	FLUX.2 AE	FLUX.2 AE	FLUX.2 AE
Class conditional	✓	–	✓	✓	✓
Number of GPUs	8	8	4	8	8
Duration (Mimg)	200	80	3000	2000	2000
Warmup n_w (Mimg)	1	1	10	10	10
Burn-in n_b (Mimg)	20	20	200	200	200
Minibatch size	512	256	4096	4096	4096
Learning rate: $\eta_\mathrm{max} \to \eta_\mathrm{ref}$	$0.01 \to 0.002$	$0.01 \to 0.0015$	$0.01 \to 0.0038$	$0.01 \to 0.0024$	$0.01 \to 0.0019$
Learning rate: m_η	8	8	4	7	8
Learning rate: N_ref	23,000	20,000	250,000	250,000	250,000
γ_{R_1}: $\gamma_0 \to \gamma_1$	$0.00325 \to 0.013$	$12.5 \to 50$	$0.5 \to 2$	$0.8 \to 3.2$	$1.5 \to 6$
γ_{R_2}	0	0	0	0	0
Adam β_2: $\beta_{2,0} \to \beta_{2,1}$	$0.9 \to 0.99$	$0.9 \to 0.99$	$0.9 \to 0.99$	$0.9 \to 0.99$	$0.9 \to 0.99$
EMA σ_rel	4.5%	3%	8.5%	7%	9.5%
Channels per resolution	512-512-512	192-384-768-768-768-768	512-512-512	1024-1024-1024	1536-1536-1536
ResBlocks per resolution	4-4-4	2-2-2-2-2-2	4-4-4	4-4-4	4-4-4
ResBlock expansion ratio	2	2	2	2	2
3×3 conv group size	32	32	32	32	32
G noise dimension	64	64	64	64	64
G class emb dimension	64	–	64	64	64
D class emb dimension	512	–	512	1024	1536
G Mparams	17.34	26.65	17.41	60.98	130.76
D Mparams	17.72	26.54	18.24	63.75	136.52
Dataset x-flips	✓	✓	✓	✓	✓
Augment probability: $p_0 \to p_1$	$0 \to 0.55$	$0 \to 0.25$	$0 \to 0.3$	$0 \to 0.4$	$0 \to 0.4$
Probability multipliers	`xflip`: 1 `rotate90`: 1 `xint`: 1 `scale`: 1 `rotate`: 1 `aniso`: 1 `xfrac`: 1 `brightness`: 0.5 `contrast`: 0.5 `lumaflip`: 0.5 `hue`: 0.5 `saturation`: 0.5 `cutout`: 1	`xflip`: 1 `rotate90`: 1 `xint`: 1 `scale`: 1 `rotate`: 1 `aniso`: 1 `xfrac`: 1 `brightness`: 0.5 `contrast`: 0.5 `lumaflip`: 0.5 `hue`: 0.5 `saturation`: 0.5 `cutout`: 1	`rotate90`: 1 `xint`: 1 `scale`: 1 `rotate`: 1 `aniso`: 1 `xfrac`: 1 `cutout`: 1	`rotate90`: 1 `xint`: 1 `scale`: 1 `rotate`: 1 `aniso`: 1 `xfrac`: 1 `cutout`: 1	`rotate90`: 1 `xint`: 1 `scale`: 1 `rotate`: 1 `aniso`: 1 `xfrac`: 1 `cutout`: 1

second-moment estimate changes at a constant relative rate:

$$\tau_i = -\frac{1}{\log \beta_{2,i}}, \quad i \in \{0, 1\}, \qquad \tau(n) = \tau_0 \left(\frac{\tau_1}{\tau_0}\right)^{u(n)}, \qquad \beta_2(n) = \exp\left(-\frac{1}{\tau(n)}\right). \tag{57}$$

For the learning rate, we specify the peak value η_max reached at the end of warmup, the reference value η_ref reached at the end of burn-in, and an initial decay multiplier m_η. During burn-in, the learning rate decreases from η_max to η_ref, with its initial rate of decrease set to m_η times the average over this interval. We construct this transition to match both the value and derivative of the subsequent inverse-square-root decay. The complete schedule is

$$\eta(n) = \begin{cases} \eta_\mathrm{max} \dfrac{n}{n_\mathrm{w}}, & 0 \le n < n_\mathrm{w}, \\ \eta_\mathrm{ref} + \Delta\eta\, v\left[s + (1-s)v^a\right], & n_\mathrm{w} \le n < n_\mathrm{b}, \\ \dfrac{\eta_\mathrm{ref}}{\sqrt{1 + (n - n_\mathrm{b})/n_\mathrm{d}}}, & n \ge n_\mathrm{b}, \end{cases} \tag{58}$$

where

$$\Delta\eta = \eta_{\max} - \eta_{\mathrm{ref}}, \qquad v = 1 - u(n),$$
$$s = \frac{\eta_{\mathrm{ref}}(n_{\mathrm{b}} - n_{\mathrm{w}})}{2n_{\mathrm{d}}\Delta\eta}, \qquad a = \frac{m_\eta - 1}{1 - s}. \tag{59}$$

Following EDM2 [10], we define the decay scale as $n_{\mathrm{d}} = \mathrm{BN}_{\mathrm{ref}}$, where B is the minibatch size and $\mathrm{N}_{\mathrm{ref}}$ is the reference number of minibatches. All schedule parameters are specified for each dataset and model scale in Tab. E.1.

Generator EMA. We use the power-function exponential moving average (EMA) and post-hoc reconstruction method of EDM2 [10]. During training, we maintain EMA profiles with relative standard deviations $\sigma_{\mathrm{rel}} \in \{0.050, 0.100, 0.150, 0.200\}$ for ImageNet and $\sigma_{\mathrm{rel}} \in \{0.010, 0.050, 0.100\}$ for CIFAR-10 and FFHQ. Then, we sweep σ_{rel} through post-hoc reconstruction from the saved EMA snapshots and report the lowest FID value for separate seeds in Tab. E.1.

Compute resources. Training the ImageNet S model takes approximately 5 days on 4 NVIDIA B200 GPUs. The M and L models each use 8 B200 GPUs; M takes approximately 4 days and L takes 7 days. The CIFAR-10 and FFHQ models each use 8 NVIDIA RTX PRO 6000 Blackwell GPUs; CIFAR-10 takes approximately 1 day and FFHQ 2 days.

E.2 Efficient Gradient Penalty Evaluation

Explicit discriminator VJP. The R_1 and R_2 penalties both require the squared discriminator input-gradient norm,

$$R(\psi) = \frac{\gamma}{2}\mathbb{E}_x\left[\|\nabla_x D_\psi(x)\|_2^2\right], \tag{60}$$

where the distribution of x distinguishes the two penalties. In PyTorch, the usual implementation obtains $\nabla_x D_\psi(x)$ through autograd with a retained derivative graph, and then differentiates through this backward computation when optimizing R. This requires constructing and differentiating a generic backward graph through the full discriminator, which is substantially less efficient in practice than expressing the same computation directly using ordinary differentiable tensor operations. Therefore, we construct the discriminator input gradient explicitly from module-wise VJPs. Consider a discriminator composed of

$$\mathbf{h}_i = f_i(\mathbf{h}_{i-1}), \qquad \mathbf{h}_0 \equiv x, \qquad \mathbf{h}_{\mathrm{L}} = D_\psi(x). \tag{61}$$

Starting from the gradient of the scalar final module,

$$\mathbf{v}_{\mathrm{L}-1} = \nabla_{\mathbf{h}_{\mathrm{L}-1}} f_{\mathrm{L}}(\mathbf{h}_{\mathrm{L}-1}), \tag{62}$$

we recursively propagate

$$\mathbf{v}_{i-1} = \mathbf{J}_{f_i}(\mathbf{h}_{i-1})^\top \mathbf{v}_i, \qquad i = \mathrm{L}-1, \ldots, 1. \tag{63}$$

By the chain rule,

$$\nabla_x D_\psi(x) = \mathbf{J}_{D_\psi}(x) = \mathbf{v}_0, \tag{64}$$

where $\mathbf{J}_{D_\psi}(x)$ denotes the column-form Jacobian of the scalar discriminator output. Thus, the gradient penalty can be evaluated as the ordinary scalar loss

$$R(\psi) = \frac{\gamma}{2}\mathbb{E}_x\left[\|\mathbf{v}_0\|_2^2\right]. \tag{65}$$

The objective is unchanged; the discriminator backward appearing inside the penalty has simply been lowered into an explicit differentiable computation.

Module-wise VJP implementation. Each VJP in Eq. 63 is expressed using ordinary differentiable tensor operations while retaining its dependence on the corresponding module parameters. Therefore, autograd can optimize the resulting gradient penalty using a standard first-order backward pass, recovering the same parameter gradients that would otherwise require generic higher-order autodiff. The discriminator consists of four relevant operations: residual blocks, resolution transitions, the discriminator head, and the input extraction layer.

For a residual block, consider, omitting magnitude-preserving scaling factors for clarity,

$$F(\mathbf{h}) = \mathbf{h} + \mathbf{W}_3\sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{h})), \tag{66}$$

where $\mathbf{W}_1$ and $\mathbf{W}_3$ are pointwise convolutions, $\mathbf{W}_2$ is a grouped 3×3 convolution, and σ is LReLU. Let $\mathbf{S}_1$ and $\mathbf{S}_2$ denote diagonal operators containing the corresponding LReLU derivatives. Given an incoming cotangent $\mathbf{v}$, the block VJP is

$$\mathbf{J}_F(\mathbf{h})^\top \mathbf{v} = \mathbf{v} + \mathbf{W}_1^\top \mathbf{S}_1 \mathbf{W}_2^\top \mathbf{S}_2 \mathbf{W}_3^\top \mathbf{v}. \tag{67}$$

For a downsampling transition with channel ratio $q \in \{1, 2, 4\}$, write the forward operator from App. C.3 as

$$\mathcal{R}_q = \mathcal{A}_q \mathcal{U} \mathcal{F}, \tag{68}$$

where $\mathcal{F}$ is the low-pass filter, $\mathcal{U}$ is pixel unshuffle, and $\mathcal{A}_q$ is the phase reduction $\mathrm{avg}_{4/q}$ of App. C.3. Its VJP is simply the adjoint

$$\mathcal{R}_q^\top \mathbf{v} = \mathcal{F}^\top \mathcal{U}^\top \mathcal{A}_q^\top \mathbf{v}. \tag{69}$$

For $q = 1$, this reduces to ordinary interpolation and is handled by a dedicated fast path, while $q = 2$ and $q = 4$ use the corresponding polyphase VJPs.

The discriminator head consists of an interpolation step followed by a pointwise convolution, LReLU, the grouped discriminative basis, a second LReLU, and a final linear layer. Denoting their weights by $\mathbf{W}_{h1}$, $\mathbf{W}_{h2}$, and $\mathbf{W}_{h3}$, its VJP is

$$\mathbf{v}_{\mathrm{head}} = \mathcal{R}_1^\top \mathbf{W}_{h1}^\top \mathbf{S}_{h1} \mathbf{W}_{h2}^\top \mathbf{S}_{h2} \mathbf{W}_{h3}^\top \mathbf{c}(y), \tag{70}$$

where $\mathbf{c}(y)$ is the class-projection cotangent, or the scalar output cotangent in the unconditional case. The input extraction layer is a pointwise convolution, so its VJP is simply

$$\mathbf{v}_x = \mathbf{W}_{\mathrm{ext}}^\top \mathbf{v}. \tag{71}$$

Composing these module VJPs in reverse network order yields the complete discriminator input gradient $\nabla_x D_\psi(x)$.

Composition with autograd. The explicit VJP replaces differentiation through the discriminator itself, while augmentation can remain under ordinary autograd. Let $y = A(x)$ denote an augmented input. After explicitly computing

$$\mathbf{v}_y = \nabla_y D_\psi(y), \tag{72}$$

autograd propagates this cotangent through the augmentation as

$$\nabla_x D_\psi(A(x)) = \mathbf{J}_A(x)^\top \mathbf{v}_y. \tag{73}$$

PyTorch's VJP interface directly supports this composition by taking $\mathbf{v}_y$ as the cotangent supplied through `grad_outputs`, so the multiplication by $\mathbf{J}_A(x)^\top$ is handled by ordinary autograd. Thus, explicit VJPs are required only for the discriminator; arbitrary differentiable augmentations can be composed without deriving their VJPs manually. On a B200, the explicit VJP reduces the R_1 time from 1,369.75 to 181.67 ms, a further $7.5\times$ over mixed precision alone (Tab. E.2).

E.3 Mixed Precision and Numerical Handling

Numerical format. Our models train successfully with either FP16 or BF16, and we observe no meaningful difference in optimization behavior or sample quality between the two. FP16, however, requires loss scaling to maintain sufficient dynamic range during backpropagation, whereas BF16 can be used directly. Therefore, we use BF16 for all final models for implementation convenience. BF16 is not our default format: we keep FP32 wherever it costs little, and use BF16 only for large activation and cotangent tensors and for tensor-core operands, where lower precision cuts memory traffic or enables fast matrix multiplication and convolution.

Explicit precision control. We implement mixed-precision training explicitly rather than relying on framework-level automatic precision selection. In particular, we disable AMP/autocast, TF32, and reduced-precision reductions, and require operations designated as FP32 to use IEEE FP32 arithmetic. Therefore, precision transitions are defined by the individual network modules rather than inferred by the training framework.

Within the main residual trunks of G and D, persistent activations and propagated cotangents are stored in BF16, and the large matrix-multiplication and convolution operands are likewise BF16. Their tensor-core products accumulate in FP32, while parameter gradients are produced in FP32. All pointwise operations, including LReLU, residual addition, fixed and learned gains, and generator noise injection, are evaluated in FP32. Retaining FP32 for these operations is inexpensive because they are

either fused into the surrounding CUDA kernels or folded into adjacent linear weights; no additional activation-sized FP32 tensors need to be materialized. The explicit discriminator VJPs in App. E.2 follow the same precision boundaries as ordinary backpropagation, with BF16 cotangents between trunk modules and the same FP32 local arithmetic.

We keep the network boundaries in FP32 until they enter, or after they leave, the large BF16 trunks. In G, the head operates in FP32 and the first residual block performs the FP32-to-BF16 transition; the remaining trunk is BF16, while the final aggregation layer produces the generator output directly in FP32. In D, the input remains FP32 until the extraction layer produces the BF16 trunk representation. Symmetrically, the final residual block produces FP32 output for the discriminator head, whose basis and class-projection operations remain in FP32.

Resolution transitions follow the same principle. Their stored inputs and outputs are BF16 when they lie inside the trunk, but channel gains, filtering, and phase aggregation are evaluated in FP32 before a single BF16 output rounding. Master parameters, parameter gradients, optimizer states, and optimizer updates remain in FP32 throughout training. On a B200, mixed precision alone makes the training step $6.7\times$ faster than the FP32 baseline (Tab. E.2).

E.4 Efficient CUDA Implementation

Our CUDA implementation was developed for NVIDIA SM100 GPUs and subsequently backported to earlier architectures. On a B200, the custom kernels reduce the training-step time from 1,024.96 to 318.38 ms, providing a further $3.2\times$ speedup relative to the explicit VJP implementation (Tab. E.2). We describe the SM100 implementation below.

Weight preparation. A single kernel applies the prescribed weight centering and row-wise ℓ_2 normalization to the three block weights, folds β_i into $\mathbf{W}_1$ and α_i into $\mathbf{W}_3$ (Eq. 6), and absorbs the magnitude-preserving LReLU gains into $\mathbf{W}_2$ and $\mathbf{W}_3$. These transformations are evaluated in FP32, producing BF16 weights while retaining the bias and noise-scale coefficients in FP32. The $\mathbf{W}_2$ coefficients are initially written in canonical order, then rearranged into the operand layouts required by the grouped-convolution kernels through separate packing operations.

Pointwise convolutions. We implement the pointwise convolutions $\mathbf{W}_1$ and $\mathbf{W}_3$ as dense tensor-core GEMMs by flattening the batch and spatial dimensions. For the interior BF16 blocks, the CUTLASS kernels use TMA-based, warp-specialized mainloops. Matrix tiles and epilogue subtiles are selected by fixed rules based on the GEMM variant, flattened row count, and channel dimensions. The fused epilogues perform pointwise operations in FP32 before converting the outputs to BF16: $\mathbf{W}_1$ applies bias, optional noise injection, and LReLU, while $\mathbf{W}_3$ adds the residual. This fusion avoids intermediate feature-map traffic and separate pointwise kernel launches. Noise injection uses one noise value per batch-spatial position, broadcast across channels with per-channel scale coefficients, without materializing an activation-sized noise tensor. For supported shapes, the $\mathbf{W}_1$ epilogue also packs the signs of its BF16 outputs into a mask containing one bit per activation. These sign bits are used during the backward pass to select LReLU slopes without reading the full activation values for that purpose.

At the FP32 boundaries of the residual network, the corresponding $\mathbf{W}_3$ projections instead use cuBLAS GEMMs with BF16 operands and FP32 outputs, allowing the residual addition to be performed in FP32. The matching $\mathbf{W}_1^\top$ operations in the backward pass likewise accumulate the skip-path cotangent in FP32.

Grouped convolution. The grouped 3×3 convolution $\mathbf{W}_2$ has 32 input channels and 32 output channels per group. In the weight-stationary implementation, each persistent CTA loads the packed weights for a pair of channel groups once from global memory and reuses them across spatial tiles. As the CTA advances through tiles, TMA stages the required input neighborhoods in shared-memory panels. Tap-dependent operand offsets select the activation operands for the nine spatial taps from these panels, avoiding explicit im2col materialization. Separate warps coordinate input loading, MMA issue, and epilogue processing, while a tensor-core collector buffer enables operand reuse across spatial taps. Multiple FP32 accumulator sets in TMEM allow accumulation for one tile to overlap with epilogue processing of the preceding tile. The epilogue reads accumulated values into registers, applies LReLU in FP32, and converts the results to BF16 for storage. Once the epilogue has completed its reads from an accumulator set, that set can be reused without waiting for output writeback to finish. When enabled, the epilogue also emits a packed sign mask for the second LReLU stage. Other supported spatial configurations use specialized kernel variants with different tiling and group assignments.

Backward pass. Activation-cotangent propagation follows the residual branch in reverse, with the intervening pointwise operations fused into the three contractions. The $\mathbf{W}_3^\top$ GEMM applies the second LReLU derivative, the grouped $\mathbf{W}_2^\top$ kernel applies the first, and the $\mathbf{W}_1^\top$ GEMM adds the skip-path cotangent. The grouped $\mathbf{W}_2^\top$ contraction reuses the forward grouped-convolution implementation, with each group's channel matrix transposed, its spatial taps reversed, and the epilogue modified to apply the LReLU derivative. The required slopes depend only on the signs of the forward BF16 activations. The $\mathbf{W}_3^\top$ epilogue obtains these signs from the packed mask emitted by $\mathbf{W}_2$, while the grouped $\mathbf{W}_2^\top$ epilogue uses the mask emitted by $\mathbf{W}_1$. Where packed masks are unavailable, the corresponding epilogues instead read the saved BF16 activations.

Separate contraction kernels compute weight gradients by reducing over batch elements and output spatial positions. The $\mathbf{W}_1$ and $\mathbf{W}_3$ gradients use cuBLAS GEMMs with BF16 operands and FP32 outputs. The grouped $\mathbf{W}_2$ gradient instead uses a custom dense tensor-core kernel that follows the forward convolution's approach to shared-memory staging and shifted-window operand selection. Unlike the forward convolution, each CTA processes one channel group over a subset of spatial tiles, staging both activation and cotangent tiles in swizzled shared memory. Tensor-core MMAs accumulate partial weight gradients in FP32 TMEM. These partial results are then combined in a fixed order without floating-point atomics. This reduction writes the result directly in the final $[C, 32, 3, 3]$ parameter layout, avoiding a separate layout conversion. Dedicated FP32 reductions compute bias and, where applicable, noise-scale gradients, reusing vectorized cotangent reads for both quantities. Gradients with respect to the prepared weights are then propagated through the folded gains, weight centering, and row-wise normalization.

The explicit discriminator VJP of App. E.2 reuses these kernels for its evaluation and differentiation. During VJP differentiation, the grouped convolution-and-slope operation uses the forward $\mathbf{W}_2$ operand rather than the transposed and spatially reversed operand used for $\mathbf{W}_2^\top$. Ordinary backward and explicit VJP evaluation can reuse the same packed activation-sign masks associated with the two saved LReLU stages.

Network heads. The 1×1 convolutions in the heads of G and D are evaluated as dense FP32 GEMMs after flattening the batch and spatial dimensions. The discriminator basis admits a similar lowering: its grouped convolution spans the full spatial field without padding, producing a single spatial location for each output channel. Therefore, flattening each group's channels and spatial dimensions gives an independent dense matrix multiplication, which we evaluate using a batched GEMM rather than separate convolution launches for each group. For sufficiently long full-field contractions, the forward computation partitions the reduction into smaller GEMMs and combines their partial results in FP32. The corresponding transposed products implement activation-cotangent propagation and the explicit discriminator VJP, while weight gradients follow the same GEMM formulation.

Resampling. Resolution transitions fuse low-pass filtering with the associated spatial and channel rearrangements. For the fixed-channel case ($q = 1$), $2\times$ upsampling or downsampling is combined with a 4×4 low-pass filter in a single kernel. For channel ratios $q \in \{2, 4\}$, specialized polyphase kernels fuse 3×3 filtering with pixel shuffle or unshuffle, incorporating the phase duplication or averaging required for $q = 2$ without materializing intermediate feature maps. Channel-wise gains are applied within the same kernels. Inputs are loaded in 16-byte channel vectors and staged in shared memory, allowing neighboring output positions to reuse their filter neighborhoods. Filtering and gain application are performed in FP32 before output conversion. Activation-cotangent propagation and the explicit discriminator VJP use adjoint variants of the same resampling operators. Where applicable, channel-gain gradient contributions are accumulated alongside resampling and combined through fixed-order FP32 reductions.

E.5 Training Throughput

Benchmark setup. We evaluate compute-only training throughput for our ImageNet S model (the CONFIG D architecture, with R_1 only, as in all final models) on a single GPU at an effective batch size of 1024. We use fixed synthetic $32 \times 32 \times 32$ latent inputs and fixed generator noise, all stored on the GPU. Each step consists of a discriminator update with R_1 regularization followed by a generator update, both including Adam and forced post-update weight normalization. For the implementation (Tab. E.2) and hardware (Tab. E.3) comparisons, each configuration uses its largest fitting power-of-two microbatch and accumulates gradients as needed to maintain the effective batch size; the batch column lists accumulation rounds $\times$ microbatch size. We measure 50 complete steps using CUDA events after 10 warmup steps and report the median duration, excluding data loading, encoding, augmentation, EMA updates, and compilation. Peak memory denotes PyTorch-allocated CUDA memory, and speedups

Table E.2: **Each optimization compounds: the step becomes 60× faster and R_1 209× cheaper.** Training throughput across implementations. D is the discriminator update without R_1, G the generator update, and R_1 the additional discriminator time required for R_1 regularization. VJP recomputation is disabled.

		Peak mem.	Total step		D	G	R_1		
Implementation (B200)	Batch	(GiB)	(ms)	Speedup	(ms)	(ms)	(ms)	(% step)	Speedup
Naive FP32 PyTorch	2×512	101.24	19,087.89	1.00×	3,421.13	2,992.03	12,674.27	66.40	1.00×
+ Mixed precision	1×1024	111.97	2,850.39	6.70×	763.96	716.62	1,369.75	48.05	9.25×
+ Explicit VJP	1×1024	106.50	1,024.96	18.62×	457.39	386.37	181.67	17.72	69.77×
+ CUDA kernels	1×1024	86.02	318.38	59.95×	138.74	119.27	60.58	19.03	209.21×

Table E.3: **Training speedups range from 14× on RTX A6000 to 60× on B200.** Performance across GPU models; speedups are relative to naive PyTorch on the same GPU. B200 uses the updated optimized implementation; other GPUs use the earlier implementation. VJP recomputation is disabled throughout. R_1 accounts for approximately 19% of the step on every GPU.

		Peak mem.	Step		R_1
GPU	Batch	(GiB)	(ms)	Speedup	(% step)
RTX A6000	4×256	23.03	2,972.54	13.96×	18.53
L40S	4×256	23.03	1,938.24	14.15×	18.97
RTX PRO 6000 Blackwell	1×1024	89.45	1,526.50	17.12×	18.38
A100 80GB	2×512	45.18	1,717.10	24.47×	18.88
H100 80GB	2×512	45.24	592.49	30.59×	19.12
B200	1×1024	86.02	318.38	59.95×	19.03

are relative to the naive PyTorch baseline on the same GPU model. All measurements use PyTorch 2.11.0+cu128, CUDA runtime 12.8, cuDNN 9.19.0, and NVIDIA driver 575.57.08, with TF32 disabled.

F Additional Results

Table F.2 extends Tab. 5 with the full set of ImageNet 256×256 baselines we considered, and Fig. F.1 breaks down where each method's inference FLOPs go. Table F.3 compares CIFAR-10 results, and Tab. F.4 FFHQ results. Qualitative samples on ImageNet (Fig. F.2), FFHQ (Fig. F.3), and CIFAR-10 (Fig. F.4), and per-class ImageNet comparisons (Figs. F.5 to F.9) close the section. Table F.1 reports further metrics.

F.1 Evaluation Protocol

Every ImageNet 256×256 number we measured, for our models and for each re-evaluated baseline, follows one protocol. A run generates 50,000 images with exactly 50 per class; its class labels and noise are fixed by its seed, and every model is evaluated with the same five seeds (100–104). We report the mean over the five runs, and the appendix tables add the sample standard deviation. FID [46] uses Inception-v3 features against ADM's reference statistics [47]; $\mathrm{FD}_{\mathrm{DINOv2}}$ uses DINOv2 ViT-L/14 features, computed with dgm-eval [48], against its 50,000-image ImageNet reference set; precision and recall use ADM's evaluator (Inception features, $k = 3$) against its 10,000-image reference batch [47, 81]. Baselines are evaluated from their released checkpoints with the sampling settings their papers report; values marked † are published, under each paper's own protocol. On CIFAR-10, runs use exactly 5,000 images per class; FFHQ 256×256 is unconditional. For both, FID is computed against the full training set (50,000 and 70,000 images), and FFHQ's $\mathrm{FD}_{\mathrm{DINOv2}}$ against dgm-eval's 50,000-image FFHQ reference set.

Table F.1: **Every metric improves from R3GAN2-S to R3GAN2-L.** Further metrics for our models; $\text{FD}_{\text{DINOv2}}$ is in Tabs. F.2 and F.4 and Inception precision and recall in Tab. 6.

- 50k samples; mean ± sd over five seeds; the same runs as those tables, except ImageNet CLIP FD (‡).
- **FID**, **KID** ($\times 10^3$): Inception features, against the training set; for ImageNet this reference is all training images, not ADM's statistics, so FID differs slightly from Tab. 5. **CLIP FD**: CLIP ViT-B/32 features; ‡ uniformly random class labels instead of 50 per class; not computed for CIFAR-10. **Precision**, **recall**: VGG16 features [81].

Model	FID (train)↓	KID↓	CLIP FD↓	Prec.↑	Recall↑
CIFAR-10	1.65 ± 0.01	0.090 ± 0.012	–	0.691 ± 0.001	0.569 ± 0.002
FFHQ-256	2.19 ± 0.03	0.473 ± 0.017	2.25 ± 0.01	0.684 ± 0.002	0.526 ± 0.001
ImageNet-256					
R3GAN2-S	2.07 ± 0.05	0.345 ± 0.022	$1.93^{\ddagger} \pm 0.01$	0.684 ± 0.003	0.635 ± 0.002
R3GAN2-M	1.55 ± 0.02	0.210 ± 0.009	$1.48^{\ddagger} \pm 0.00$	0.709 ± 0.001	0.650 ± 0.002
R3GAN2-L	1.47 ± 0.00	0.184 ± 0.006	$1.44^{\ddagger} \pm 0.01$	0.714 ± 0.002	0.655 ± 0.002

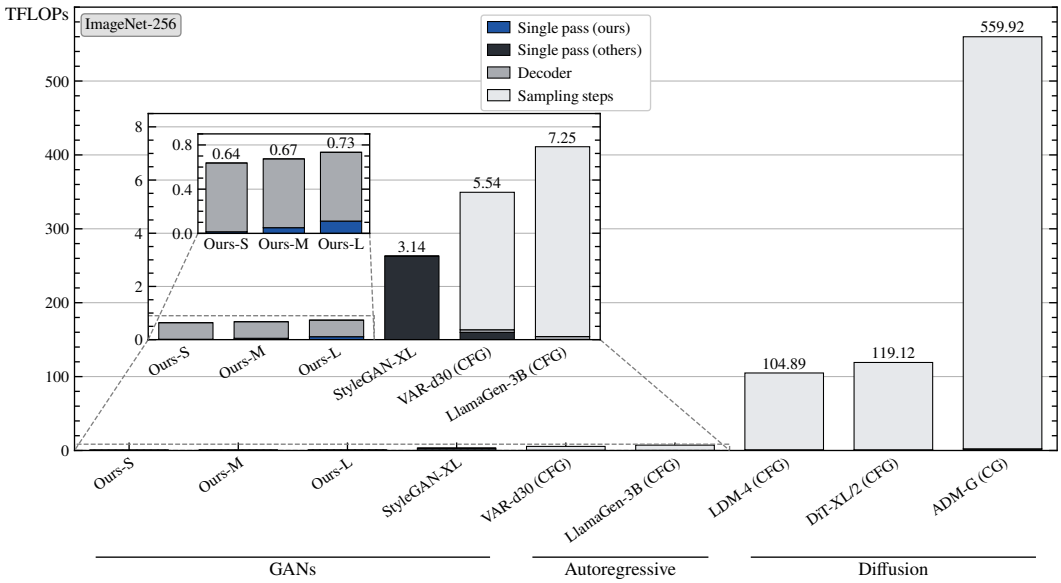


Figure F.1: **Our models are FLOP-efficient versus autoregressive and diffusion models.** Inference TFLOPs per generated ImageNet 256 × 256 image, split into one pass through the trained network, the decoder (VAE, or VQ for the AR models and LDM-4), and the remaining sampling steps; each zoom magnifies the dashed region below it. Costs follow Tab. F.2: CFG counts two forwards per step, and CG counts the generator only. For AR models, every step after the first counts as a sampling step. Appendix E.5 measures training cost.

Table F.2: **R3GAN2 efficiently reaches low FID without guidance.** Class-conditional ImageNet 256×256; Tab. 5 is a subset.

- **FD**: 50k samples; mean $\pm$ sd over five seeds; † published under paper's protocol; '–': none or not reported.
- **Feat.**: additional pretrained features in training, which can flatter that family's FD [49]. **I**: supervised ImageNet classifier, like FID's Inception; **D**: DINO family, like $\text{FD}_{\text{DINOv2}}$'s. AE training (Domain) may also use them.
- **Guide**: guidance reshapes the samples' distribution away from the model's learned conditional distribution. **CFG**: classifier-free guidance; **AG**: AutoGuidance; **CG**: classifier guidance from a classifier (rejection or gradient); subscript **t**: trained in, no sampling cost.
- **TFLOPs**: per image, including guidance and decoding; a multiply-add counts as 2 FLOPs; '$\approx$': estimated from the architecture (KV-cache AR: $\approx 2\times\text{params}\times\text{tokens}$; CG: generator only).
- **Notes**: iCT-XL/2 is IMM's re-implementation [82]. Simple diffusion's FIDs are on the training split, its 2.44 is the best of a guidance sweep, and its NFE and TFLOPs are estimates.

								FD↓, unguided		FD↓, guided	
Method	Params	Domain	Feat.	Guide	NFE		TFLOPs	Incept.	DINOv2	Incept.	DINOv2
Autoregressive											
JetFormer-L [83]	2,750 M	flow tok.	–	CFG	256×2	≈	2.20	–	–	6.64†	–
MaskGIT [84]	227 M	VQ-VAE	–	–	8	≈	0.98	6.18†	–	–	–
RQ-Transformer [85]	3,800 M	RQ-VAE	–	CG	256×8	≈	6.00	7.55†	–	3.80†	–
STARFlow [86]	1,400 M	SD-VAE	–	CFG	1,024×2	≈	6.36	–	–	2.40†	–
LlamaGen-3B [87]	3,100 M	VQ-VAE	–	CFG	576×2	≈	7.25	9.38†	–	2.18†	–
VAR-d30 [36]	2,000 M	MS-VQ	–	CFG	10×2	≈	5.54	–	–	1.95 $\pm$ 0.01	79.0 $\pm$ 0.3
MAR-H [6]	943 M	KL-16	–	CFG	256×2	≈	189.52	2.35†	–	1.55†	–
RAR-XXL [50]	1,500 M	MaskGIT-VQ	–	CFG	256×2	≈	1.73	–	–	1.49 $\pm$ 0.02	160.9 $\pm$ 0.6
xAR-H [88]	1,100 M	SD-VAE	–	CFG	200×2	≈	56.94	–	–	1.24†	–
Diffusion											
ADM-G [47]	554 M	Pixel	–	CG	250	≈	559.92	10.94†	–	4.59†	–
LDM-4 [63]	400 M	LDM-f4 AE	–	CFG	250×2	≈	104.89	10.56†	–	3.60†	–
JLT-B/1 [51]	130 M	FLUX.2 AE	–	CFG	99×2	≈	9.75	16.12 $\pm$ 0.09	367.3 $\pm$ 1.2	2.96 $\pm$ 0.03	160.9 $\pm$ 0.9
Diffusion (U-ViT 2B) [89]	2,000 M	Pixel	–	CFG	1,000×2	≈	4000.00	2.77†	–	2.44†	–
DiT-XL/2 [25]	675 M	SD-VAE	–	CFG	250×2	≈	119.12	9.62†	–	2.27†	–
SiT-XL/2 [52]	675 M	SD-VAE	–	CFG	250×2	≈	119.12	8.26†	–	2.06†	–
SiT-XL/2+REPA [52, 53]	675 M	SD-VAE	D	CFG	250×2	≈	100.64	5.90†	–	1.42†	–
SiD2 [90]	≈230 M	Pixel	–	CFG	512×2	≈	1337.00	–	–	1.38†	–
LightningDiT-XL/1 [91]	675 M	VA-VAE	–	CFG	250×2	≈	119.12	2.17†	–	1.35†	–
DDT-XL/2 [92]	675 M	SD-VAE	D	CFG	250×2	≈	119.12	6.30†	–	1.31†	–
RAE [54]	839 M	DINO-B	–	AG	50×2	≈	18.00	1.51 $\pm$ 0.02	48.0 $\pm$ 0.1	1.15 $\pm$ 0.01	35.9 $\pm$ 0.3
Flow map											
iCT-XL/2 [82, 93]	675 M	SD-VAE	–	–	1	≈	0.86	34.24†	–	–	–
Shortcut-XL/2 [94]	676 M	SD-VAE	–	CFG_{t}	1	≈	0.86	–	–	10.60†	–
MeanFlow-XL/2 [12]	676 M	SD-VAE	–	CFG_{t}	1	≈	0.86	–	–	3.43†	–
iMF-B/2 [56]	89 M	SD-VAE	–	CFG_{t}	1	≈	0.67	–	–	3.39†	–
TiM-XL/2 [95]	664 M	SD-VAE	D	–	1	≈	0.86	6.95 $\pm$ 0.11	260.2 $\pm$ 1.2	–	–
TiM-XL/2 (MG) [95]	664 M	SD-VAE	D	CFG_{t}	1	≈	0.86	–	–	3.26†	–
α-Flow-XL/2+ [55]	676 M	SD-VAE	–	CFG_{t}	1	≈	0.86	–	–	2.46 $\pm$ 0.01	152.7 $\pm$ 0.6
iMF-M/2 [56]	174 M	SD-VAE	–	CFG_{t}	1	≈	0.72	–	–	2.27†	–
iMF-L/2 [56]	409 M	SD-VAE	–	CFG_{t}	1	≈	0.85	–	–	1.86†	–
iMF-XL/2 [56]	610 M	SD-VAE	–	CFG_{t}	1	≈	0.97	10.61 $\pm$ 0.10	272.3 $\pm$ 0.9	1.82 $\pm$ 0.02	105.4 $\pm$ 0.5
iCT-XL/2 [82, 93]	675 M	SD-VAE	–	–	2	≈	1.10	20.30†	–	–	–
IMM-XL/2 [82]	675 M	SD-VAE	–	CFG	1×2	≈	1.10	–	–	7.93 $\pm$ 0.09	252.6 $\pm$ 1.0
MeanFlow-XL/2+ [12]	676 M	SD-VAE	–	CFG_{t}	2	≈	1.10	–	–	2.20†	–
α-Flow-XL/2+ [55]	676 M	SD-VAE	–	CFG_{t}	2	≈	1.10	–	–	2.06 $\pm$ 0.02	101.0 $\pm$ 0.8
iMF-XL/2 [56]	610 M	SD-VAE	–	CFG_{t}	2	≈	1.32	–	–	1.61 $\pm$ 0.02	91.9 $\pm$ 0.4
Distilled flow map											
π-Flow-XL/2 (GM-REPA) [96]	675 M	SD-VAE	D	CFG_{t}	1	≈	0.86	–	–	2.69 $\pm$ 0.05	114.8 $\pm$ 0.5
FACM-XL/2 [57]	675 M	VA-VAE	–	CFG_{t}	1	≈	0.86	–	–	1.72 $\pm$ 0.01	82.4 $\pm$ 0.3
π-Flow-XL/2 (GM-REPA) [96]	675 M	SD-VAE	D	CFG_{t}	2	≈	1.10	–	–	1.83 $\pm$ 0.02	86.2 $\pm$ 0.5
FACM-XL/2 [57]	675 M	VA-VAE	–	CFG_{t}	2	≈	1.10	–	–	1.34 $\pm$ 0.00	65.4 $\pm$ 0.2
GAN											
BigGAN-deep [59]	56 M	Pixel	–	–	1		0.08	6.89 $\pm$ 0.04	399.7 $\pm$ 1.2	–	–
GigaGAN [44]	569 M	Pixel	–	–	1	≈	10.77	3.45†	–	–	–
StyleGAN-XL [24]	166 M	Pixel	I	CG_{t}	1		3.14	–	–	2.34 $\pm$ 0.02	214.9 $\pm$ 0.5
StyleSAN-XL [58]	166 M	Pixel	I	CG_{t}	1		3.14	–	–	2.27 $\pm$ 0.01	224.1 $\pm$ 0.9
GAT-XL/2 [62]	602 M	SD-VAE	D	–	1	≈	0.83	2.18†	–	–	–
R3GAN2-S (ours)	17 M	FLUX.2 AE	–	–	1		0.64	2.12 $\pm$ 0.05	163.9 $\pm$ 1.0	–	–
R3GAN2-M (ours)	61 M	FLUX.2 AE	–	–	1		0.67	1.60 $\pm$ 0.01	125.7 $\pm$ 0.7	–	–
R3GAN2-L (ours)	131 M	FLUX.2 AE	–	–	1		0.73	1.52 $\pm$ 0.00	115.1 $\pm$ 0.5	–	–

Table F.3: **R3GAN2 has the lowest FID of any model without ImageNet features.** CIFAR-10, pixel space.

- **FID**: 50k samples; mean ± sd over five seeds; † published under paper's protocol.
- **Feat.**: additional pretrained features in training, which can flatter that family's FD [49]. **I**: supervised ImageNet classifier, like FID's Inception. **Guide**: $\mathbf{CG_t}$: classifier guidance trained in; '–': none.
- **Notes**: our model is class-conditional; the lower block's methods report only unconditional results. Conditioning improves CIFAR-10 FID only modestly (as published, conditional vs. unconditional: EDM 1.79 vs. 1.97; StyleGAN2 + ADA 2.42 vs. 2.92): a label carries at most $\log_2 10 \approx 3.3$ bits about its image, against ≈ 10 on ImageNet. StyleGAN3-R + ADA is StudioGAN's re-implementation [97]. Params (generator) and GFLOPs (total per sample) are our counts, as in Tab. F.2; '≈': estimated.

Method	Type	Params	Feat.	Guide	NFE	GFLOPs	FID↓
Class-conditional							
VE [9, 98]	Diffusion	63 M	–	–	35	770.7	3.15 ± 0.04
VP [9, 98]	Diffusion	62 M	–	–	35	759.3	2.52 ± 0.02
EDM [9]	Diffusion	56 M	–	–	35	1,483.4	1.81 ± 0.01
BigGAN [59]	GAN	9 M	–	–	1	≈ 7.6	14.73†
StyleGAN3-R + ADA [60, 97]	GAN	24 M	–	–	1	102.7	10.83†
StyleGAN2 + ADA [41]	GAN	20 M	–	–	1	8.0	2.40 ± 0.02
R3GAN [14]	GAN	21 M	–	–	1	14.1	2.00 ± 0.02
StyleGAN-XL [24]	GAN	18 M	I	CG_t	1	69.7	1.88 ± 0.01
R3GAN2 (ours)	GAN	17 M	–	–	1	14.5	1.65 ± 0.01
StyleSAN-XL [58]	GAN	64 M	I	CG_t	1	243.3	1.39 ± 0.01
Unconditional							
DDIM [99]	Diffusion	36 M	–	–	50	622.2	4.67†
DDPM [8]	Diffusion	36 M	–	–	1,000	12,443.7	3.17†
MeanFlow [12]	Flow-map	56 M	–	–	1	42.4	2.92†
IMM [82]	Flow-map	56 M	–	–	2	84.8	2.01 ± 0.02
FACM [57]	Flow-map distill	56 M	–	–	1	≈ 42.4	2.69†
FACM [57]	Flow-map distill	56 M	–	–	2	≈ 84.8	1.87†
TransGAN [100]	GAN	83 M	–	–	1	10.8	9.26†
ViTGAN [101]	GAN	13 M	–	–	1	≈ 2.6	4.92†
DDGAN [102]	GAN	48 M	–	–	4	56.4	3.75†
Diffusion StyleGAN2 [103]	GAN	20 M	–	–	1	8.0	3.19†

Table F.4: **Without pretrained features, R3GAN2 matches StyleGAN-XL on FID.** FFHQ 256×256, unconditional.

- **FD**: 50k samples; mean $\pm$ sd over five seeds; † published under paper's protocol; '–': not reported.
- **Feat.**: additional pretrained features in training, which can flatter that family's FD [49]. **I**: supervised ImageNet classifier, like FID's Inception. AE training (Domain) may also use them.
- **Notes**: published FIDs as compiled in R3GAN [14] (its Tab. 4); ADM and Diffusion Autoencoder are compsampling's measurements [104], and its ADM is a model compsampling trained. R3GAN gives no source for the StyleGAN3 values, and StyleGAN3 released only FFHQ-U models at this resolution. Params (sampling networks, excluding a latent model's autoencoder) and TFLOPs (per sample, including decoding) are our counts, as in Tab. F.2.

							FD↓	
Method	Type	Params	Domain	Feat.	NFE	TFLOPs	Incept.	DINOv2
ADM (DDIM) [47, 104]	Diffusion	119 M	Pixel	–	500	248.52	8.41†	–
ADM (DPM-Solver) [47, 104]	Diffusion	119 M	Pixel	–	500	248.52	8.40†	–
Diffusion Autoencoder [104, 105]	Diffusion	192 M	Pixel	–	500	277.27	5.81†	–
LDM [63]	Diffusion	274 M	LDM-f4 AE	–	200	41.15	4.98†	–
StyleGAN3-T [60]	GAN	23 M	Pixel	–	1	0.24	4.81†	–
StyleGAN2 [28]	GAN	25 M	Pixel	–	1	0.03	4.01 $\pm$ 0.04	421.8 $\pm$ 0.6
StyleGAN3-R [60]	GAN	16 M	Pixel	–	1	0.23	3.92†	–
R3GAN [14]	GAN	23 M	Pixel	–	1	0.06	2.76 $\pm$ 0.04	443.3 $\pm$ 1.2
PolyINR [106]	GAN	13 M	Pixel	I	1	0.35	2.72†	–
StyleGAN-XL [24]	GAN	68 M	Pixel	I	1	1.28	2.19 $\pm$ 0.05	239.4 $\pm$ 0.5
R3GAN2 (ours)	GAN	27 M	Pixel	–	1	0.18	2.19 $\pm$ 0.03	393.1 $\pm$ 0.5
StyleSAN-XL [58]	GAN	258 M	Pixel	I	1	4.91	1.70 $\pm$ 0.02	228.2 $\pm$ 0.3



Figure F.2: Qualitative ImageNet samples.



Figure F.3: Qualitative FFHQ samples.



Figure F.4: Qualitative CIFAR-10 samples.

Real images



GAT-XL/2 [62]



iMF-XL/2 [56]



R3GAN2-L



Figure F.5: Qualitative comparison on the ImageNet *pot* class.

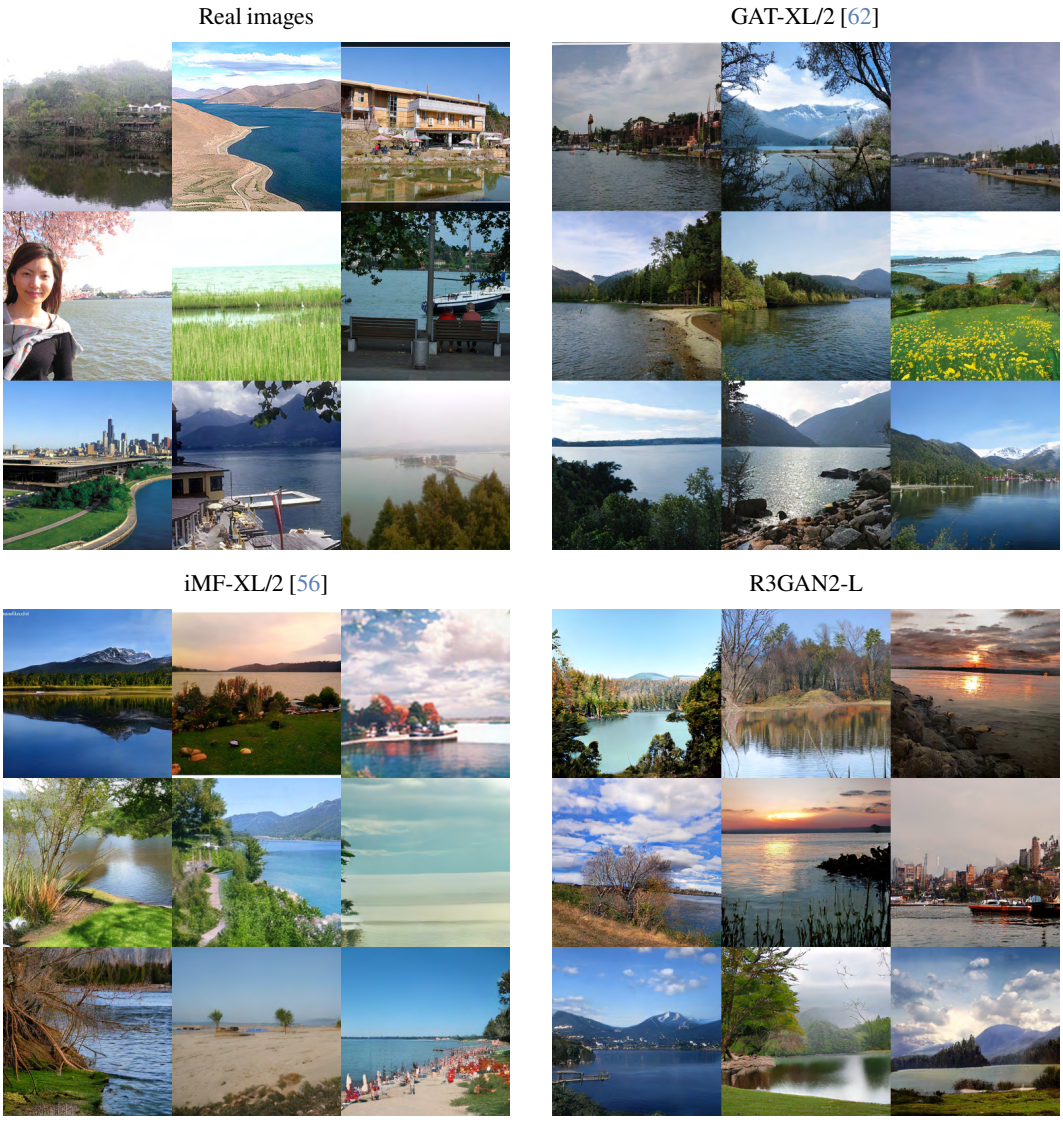


Figure F.6: Qualitative comparison on the ImageNet *lakeside* class.

Real images



GAT-XL/2 [62]



iMF-XL/2 [56]



R3GAN2-L



Figure F.7: Qualitative comparison on the ImageNet *beacon* class.

Real images

GAT-XL/2 [62]



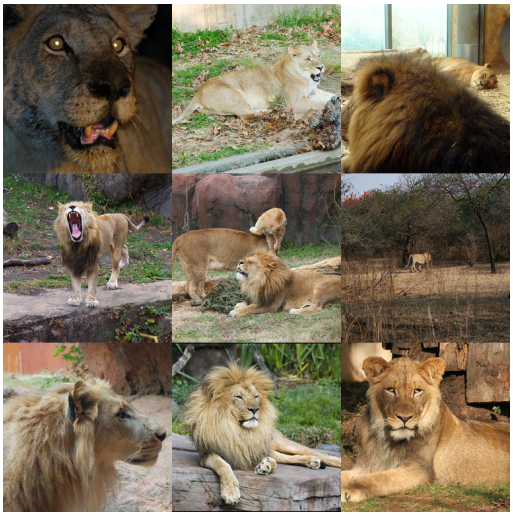
iMF-XL/2 [56]

R3GAN2-L



Figure F.8: Qualitative comparison on the ImageNet *palace* class.

Real images



GAT-XL/2 [62]



iMF-XL/2 [56]



R3GAN2-L



Figure F.9: Qualitative comparison on the ImageNet *lion* class.

G Negative Results and Future Directions

Following R3GAN [14], we report additional design choices explored during development that did not enter our final models. Most are preliminary; we include them to record what did not work and to point to open questions.

Multiplicative conditioning. Feature-wise modulation, where a conditioning signal predicts per-channel scales, is widely used in generative models, including StyleGAN2 [28], EDM2 [10], and adaLN in DiT [25]. We experimented with similar modulation using the noise vector in G and the class embedding in both G and D. Applying it only to G improved our early FFHQ and CIFAR-10 experiments, but additionally modulating D substantially increased discriminator overfitting on CIFAR-10 and ultimately worsened FID. On ImageNet, even G-only modulation consistently collapsed training.

Such multiplicative conditioning is awkward under a magnitude-preserving design: an unconstrained, sample-dependent channel scale can directly change activation magnitudes and thereby circumvent the control provided by surrounding magnitude-preserving layers. EDM2 demonstrates that such modulation can nevertheless coexist successfully with magnitude-preserving components in diffusion models, but the modulation itself is not magnitude preserving. A cleaner solution may be to avoid multiplicative conditioning altogether. For example, Improved MeanFlow [56] replaces adaLN-style conditioning with in-context conditioning through additional tokens and reports improved performance. Developing similarly effective conditioning mechanisms that preserve signal magnitude remains an interesting direction for GANs.

Multiplicative FFNs. We also experimented with GLU-style feed-forward blocks [107], where two learned branches are multiplied elementwise. Used naively, these blocks consistently destabilized and collapsed training. This is unsurprising from a magnitude-preservation perspective: for a ReLU-gated block

$$y = (Ux) \odot \mathrm{ReLU}(Vx),$$

rescaling the input by s approximately rescales the output by s^2, so its amplification intrinsically depends on the incoming activation magnitude. Adding an explicit normalization immediately after the multiplication restored stable training, but substantially increased discriminator overfitting and eventually produced worse FID.

The advantage of such multiplicative interactions may also not require an explicit learned gate. Primer [108] finds that the considerably simpler squared-ReLU FFN outperforms ReGLU and SwiGLU in their Transformer language-modeling experiments, and argues that squared ReLU may capture much of their benefit without the additional gating branch. Squared ReLU, however, has the same degree-two magnitude dependence, so it is similarly unattractive for our architecture. For future work, we are more interested in increasing FFN depth while retaining magnitude-compatible, approximately degree-one nonlinearities. Compositions of piecewise-linear layers can increase representational complexity rapidly with depth [109]; this may provide a cleaner way to increase the number of feature-space partitions and directional transformations without introducing strongly magnitude-dependent operations.

Stronger anti-aliasing filters. We also experimented with substantially more aggressive low-pass filters in the resolution-transition layers, including Lanczos and Kaiser-windowed variants inspired by alias-free resampling [60]. These filters produced a modest improvement on CIFAR-10 but consistently degraded FFHQ performance, so we retain the lightweight binomial filter used in our final models.

One possible explanation is that the role of resampling differs across these settings. On CIFAR-10, every resolution stage has the same channel width, so each shuffle-based resampling layer reduces to ordinary interpolation. In our FFHQ model, channel width decreases toward higher resolutions, and the transitions instead convert excess low-resolution channel capacity into finer spatial-phase information before subsequent high-resolution processing. More aggressive low-pass filtering may suppress part of this newly exposed high-frequency evidence. We regard this explanation as tentative, but the experiments suggest that stronger anti-aliasing is not uniformly beneficial.

Removing high-resolution stages. We also explored whether expensive high-resolution stages could be removed entirely by keeping the network at low spatial resolution and exposing the final pixels only through an $8\times$ pixel shuffle. This performed disastrously on both FFHQ and ImageNet. Increasing the low-resolution model to roughly $16\times$ the parameter count still failed to match the substantially smaller model with native high-resolution stages. We further replaced the pixel shuffle representation with three levels of recursive Haar-wavelet decomposition/synthesis, providing an explicit multiscale representation of low- and high-frequency components; this performed even worse.

These results suggest that the limitation is not merely an inconvenient choice of packed low-resolution representation. Although low-resolution channels can carry information that is converted to and from fine spatial structure, learned processing at the corresponding spatial resolutions appears difficult to eliminate. Recent pixel-space diffusion models have reached a similar architectural compromise: PixelDiT [110] combines coarse patch-level processing with a separate pixel-level pathway for texture refinement, while DeCo [111] retains a lightweight pixel decoder for high-frequency detail. Therefore, rather than removing high-resolution computation entirely, a more promising direction may be to make the high-resolution stages substantially cheaper.

Augmentation-conditioned discriminator. Following augmentation-conditioned generative models such as DistAug [112] and EDM [9], we passed the parameters of each sampled augmentation directly to the discriminator as an additional condition. This did not improve final FID in our experiments. In generator-only models, the augmentation label is needed to distinguish the different transformed distributions encountered during training. GANs admit a different solution: StyleGAN2-ADA [41] shows that augmentation need not leak into the learned distribution as long as the stochastic augmentation operator remains invertible at the distribution level. Therefore, in this setting, explicitly revealing the sampled transformation to D is not required.

Generalized color augmentation. We also generalized the color augmentations used by StyleGAN2-ADA [41] from RGB images to higher-dimensional autoencoder latents. We define the first principal component across latent channels as a luma-like axis, which gives natural extensions of brightness, contrast, saturation, and luma/chroma flips to arbitrary channel dimension; hue does not admit a unique analogue when the orthogonal subspace has dimension greater than two. These latent-space color augmentations did not improve FID in our ImageNet experiments, suggesting that mathematically natural channel-space transformations may not correspond to useful nuisance variations in a learned latent representation.

Domain-agnostic corruption. We also tried replacing hand-designed image augmentations with more generic corruption processes. Following Diffusion-GAN [103], we added Gaussian noise to both real and generated samples using diffusion-style schedules, and also experimented with the noise distribution used by EDM2 [10]. In our experiments, weak corruption left the discriminator free to overfit, while sufficiently strong corruption quickly removed too much real–fake information for meaningful discriminator learning. We were unable to find a noise schedule that provided a useful intermediate regime. Randomly masking spatial elements, motivated by masked/discrete diffusion models, exhibited essentially the same behavior: small masking ratios did little to reduce overfitting, whereas larger ratios crippled discriminator learning.

Diffusion-GAN is also not a clean counterexample to this observation. Its forward corruption attenuates the clean signal while independently controlling the injected noise scale, and later work in Scale-GAN [113] isolates these effects and finds that data scaling itself accounts for a substantial part of the stabilization, with noise providing a much smaller additional benefit. Therefore, our experiments suggest that purely information-destroying corruption may have a particularly narrow useful operating range as a replacement for conventional GAN augmentation.

Reconstruction GAN. We also explored a reconstruction-oriented GAN by factorizing generation as $z \to r \to x$, where r is an image representation. In preliminary CIFAR-10 experiments, we used CLIP image embeddings as r and focused only on the $r \to x$ stage, supplying the representation of each real image directly. Therefore, each generated sample is paired with a specific real sample, so the generator is trained toward reconstructing an exact target rather than only toward the data distribution.

We tested two ways of using this pairing. First, we conditioned D directly on r. This caused severe discriminator overfitting: because the representations were effectively unique to individual training images, $D(x, r)$ could increasingly behave like a lookup table over the training set. Second, we left D unconditional but added reconstruction losses in its intermediate feature spaces, similar to discriminator feature matching [114], so that each generated sample was explicitly transported toward its corresponding real sample without revealing r to D. This avoided the lookup-table failure mode, but, for reasons that remain unclear, still did not improve FID over a conventional GAN. Thus, in our experiments, turning GAN training into a more reconstruction-like problem did not provide an advantage over ordinary distribution matching. Whether a different representation, more careful weighting of reconstruction terms, or softer form of pairing can make a reconstruction GAN effective remains open.

Attention regularization. As discussed in Sec. 4.1, adding self-attention improves FID faster early in training but plateaus sooner, with signs of discriminator overfitting. Therefore, we tried to regularize

attention more aggressively using the Lipschitz normalization of Dasoulas *et al*. [115], which rescales attention logits according to feature norms and sequence length. In our setting this normalization was too strong: the attention matrix became nearly uniform and the attention block contributed little beyond global averaging. Weakening the normalization enough to recover meaningful attention selectivity did not solve the original problem, and the discriminator still overfit later in training. We did not find an intermediate regime where attention remained useful while avoiding this failure mode.

This result suggests that simply constraining softmax sharpness may not be sufficient to make attention beneficial in the discriminator. Drifting Models [13] likewise drive generator updates with a fully convolutional feature model, a role related to that of a GAN discriminator. Together, these results suggest that a convolutional critic may suffice, and that attention may help more in the generator or under different discriminator regularization.

Positional encoding in attention. We also tested adding rotary position embedding (RoPE) [116] to the self-attention blocks, but observed no additional benefit. This is consistent with prior work on convolution–attention hybrids: CvT [117] finds that explicit positional embeddings can be removed without degrading performance once convolutional operators are integrated into the architecture. Our attention blocks are embedded inside an otherwise convolutional hierarchy, so local spatial structure is already encoded by the surrounding network and may make an additional positional mechanism unnecessary.

A related observation has recently appeared in language models. Kimi Linear [118] applies no positional encoding to its global full-attention layers, instead relying on the interleaved position-aware KDA layers to provide positional information. This suggests a more general design pattern in which global attention need not carry its own positional encoding when another operator in the architecture already provides local or positional structure.

Dataflow accelerators. A single-step convolutional generator may suit dataflow accelerators such as Cerebras's wafer-scale engines [119] and Groq's tensor streaming processor [120], which tile many processing elements over large on-chip memory. Convolutions have regular memory access and no data-dependent control flow, so they can be tiled across such a fabric efficiently. Multi-step samplers are harder to pipeline because each step depends on the previous one, and attention adds irregular, resolution-dependent memory traffic. Our generator is a single forward pass without attention, so whole resolution stages could in principle be fused into one schedule that keeps activations on chip. On ImageNet, however, the frozen FLUX.2 decoder dominates inference cost (622 of 637–734 GFLOPs across our three scales; Fig. F.1); it is also a single forward pass, so the same argument may apply to it. We expect the efficiency advantage of one-step convolutional generation over iterative models to be larger on such hardware than on GPUs, but we have not tested this.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper makes claims about the activation magnitudes of a discriminator and the eventual performance achieved within a GAN from introducing magnitude-preserving layers within the network.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Included in Sec. 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes].

   Justification: Assumptions and proofs are in Apps. A.2 and A.3 and App. B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: Architecture, training and evaluation are described in Sec. 4 and Apps. E.1 and F.1, and the code and trained models are released at https://github.com/brownvc/R3GAN2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: Code, training and evaluation scripts, and trained models are released at https://github.com/brownvc/R3GAN2; all datasets are public.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: Hyperparameters and schedules are in App. E.1 and Tab. E.1; the evaluation protocol is in App. F.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

   Justification: Measured results are the mean over five evaluation seeds, and the appendix tables give the sample standard deviation (App. F.1). This spread is over evaluation seeds of one trained model, not over training runs, which are too expensive to repeat.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: Hardware, training time and memory are in App. E.1 (Compute resources) and App. E.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes].

   Justification: To the best of our understanding, our work conforms to the code. Indeed, it significantly improves the efficiency of GANs for a particular achieved quality, which decreases the energy required for inference.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: Included in Sec. 6.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: Our work only uses existing extremely-common datasets, e.g., ImageNet, though these have limitations that have been acknowledged by the community already as they are scraped.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: ImageNet is used under its terms of access for non-commercial research; FFHQ is CC BY-NC-SA 4.0; CIFAR-10 is publicly available without a stated licence; the FLUX.2 [dev] autoencoder is used under its non-commercial licence; baseline checkpoints are used under their authors' licences. We do not redistribute any of these.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes].

Justification: We release code and trained models, documented in the repository (https://github.com/brownvc/R3GAN2).

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A].

Justification: No crowdsourcing or human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A].

Justification: No human-subjects study; no IRB approval needed.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A].

Justification: Not relevant.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.