SpeeDiff: Scalable Pixel-Anchored End-to-End Latent Diffusion Model

Supplementary Material

A. Detailed Implementation of SpeeDiff

This section provides comprehensive implementation details of SpeeDiff to ensure full reproducibility. We describe our model architectures in Appendix A.1, training configurations in Appendix A.2, and sampling setup in Appendix A.3. A complete summary of hyperparameters and model configurations is included in Tab. 14.

A.1. Model Details

A.1.1. ViT-VAE

We adopt a vision-transformer–based VAE (ViT-VAE) in place of the conventional convolutional VAE, as transformer architectures have better scalibility and offer improved compute efficiency (see Tab. 7 for comparison).

Encoder. The encoder begins with a patch-embedding layer that tokenizes each $p \times p$ patch into a hidden vector. We use p=16 for ImageNet 256×256 and p=32 for 512×512 generation. The embedded tokens are processed by a stack of standard transformer blocks equipped with 2D RoPE [19] positional encoding and SwiGLU [48] feedforward layers. A final linear projection maps the output tokens into a compact latent representation of dimension d=32.

Decoder. The decoder mirrors the encoder. It first maps the latent tokens back to the hidden dimension through a linear layer, followed by transformer blocks, and finally an unpatch-embedding layer that reconstructs pixel-space outputs. To further refine visual details, we append a single convolutional layer initialized as an identity mapping. This lightweight refinement consistently improves visual quality.

A.1.2. Refined-DiT

Our diffusion model, Refined-DiT, is based on recent architectural advances from Lightening DiT [58] and incorporates a more parameter-efficient modulation mechanism. Due to spatial downsampling introduced by the VAE, we fix the diffusion model's patch size to 1, implemented as a simple linear projection into the transformer hidden dimension. The model then applies a sequence of diffusion transformer blocks. Instead of per-block modulation, we adopt a global modulation layer combined with learnable per-block biases, following the design strategy of PixArt-α [6]. This approach reduces parameters and computational overhead while preserving generative performance.

Table 7. **Model configurations.** Comparison between our refined architecture (ViT-VAE+Refined-DiT) and the baseline in terms of parameters, FLOPs, and inference memory per image. ViT-VAE is more scalable and memory friendly in our end-to-end training framework.

Configuration	**ViT-VAE Encoder** (ours)			**ViT-VAE Decoder** (ours)			**Refined-DiT** (ours)		
	Params (M)	FLOPs (G)	Memory (M)	Params (M)	FLOPs (G)	Memory (M)	Params (M)	FLOPs (G)	Memory (M)
Base	28.98	7.41	11.0	28.95	7.41	11.78	91.38	21.79	10.29
Large	103.68	26.54	14.75	103.65	26.54	20.03	319.38	79.01	14.06
XL	218.63	55.97	19.88	218.58	55.97	28.78	672.23	167.07	19.20
Configuration	**CNN-VAE Encoder**			**CNN-VAE Decoder**			**DiT-XL/1**		
	Params (M)	FLOPs (G)	Memory (M)	Params (M)	FLOPs (G)	Memory (M)	Params (M)	FLOPs (G)	Memory (M)
-	28.42	69.01	193.31	41.42	126.43	161.62	675.15	114.42	12.41

A.2. Training Details

Algorithm 1 SpeeDiff: End-to-end pixel-anchored joint training of latent diffusion model

Require: Data p_{data}, encoder $p_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x}_0)$, decoder $\mathcal{D}_{\boldsymbol{\psi}}$, diffusion $V_{\boldsymbol{\theta}}$, two MLP adapters h_{ω_1} and h_{ω_2} (optional)
while not converged **do**
  $\mathbf{x}_0 \sim p_{\text{data}}, \quad \mathbf{z}_0 \sim p_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x}_0)$
  ▷ **VAE Reconstruction**
  $\hat{\mathbf{x}}_{\text{vae}} = \mathcal{D}_{\boldsymbol{\psi}}(\mathbf{z}_0), \quad L_{\text{VAE}} = \mathcal{L}_{\text{VAE}}(\hat{\mathbf{x}}_{\text{vae}}, \mathbf{x}_0; \boldsymbol{\phi}, \boldsymbol{\psi})$

  ▷ **Diffusion Flow Matching**
  $t \sim \mathcal{U}(0,1),\ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}); \mathbf{z}_t = (1-t)\mathbf{z}_0 + t\boldsymbol{\epsilon},\ \mathbf{v}_t = \boldsymbol{\epsilon} - \mathbf{z}_0$
  $\mathbf{v}_{\text{pred}} = V_{\boldsymbol{\theta}}(\mathbf{z}_t, t), \quad L_{\text{Diff}} = \mathcal{L}_{\text{Diff}}(\mathbf{v}_{\text{pred}}, \mathbf{v}_t; \boldsymbol{\phi}, \boldsymbol{\theta})$

  ▷ **Tweedie Pixel Reconstruction**
  $\mathbf{z}_{\text{tweedie}} = \mathbf{z}_t - t\,\mathbf{v}_{\text{pred}}, \quad \hat{\mathbf{x}}_{\text{tpr}} = \mathcal{D}_{\boldsymbol{\psi}}(\mathbf{z}_{\text{tweedie}})$
  $L_{\text{TPR}} = \mathcal{L}_{\text{TPR}}(\hat{\mathbf{x}}_{\text{tpr}}, \mathbf{x}_0; \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\theta})$

  ▷ **REPA++ (optional)**
  $\mathbf{y} = \text{VFM}(\mathbf{x}_0)$; intermediate feature $\mathbf{f}_t$ from $V_\theta(\mathbf{z}_t, t)$
  $\mathbf{y}_{\text{latent}} = h_{\boldsymbol{\omega}_1}(\mathbf{z}_0); \mathbf{y}_{\text{diff}} = h_{\boldsymbol{\omega}_2}(\mathbf{f}_t)$
  $L_{\text{REPA++}} = \mathcal{L}_{\text{Latent-REPA}}(\mathbf{y}, \mathbf{y}_{\text{latent}}; \omega_1) + \mathcal{L}_{\text{Diff-REPA}}(\mathbf{y}, \mathbf{y}_{\text{diff}}; \omega_2)$

  For **SpeeDiff (w/o REPA++)**
  $L = L_{\text{VAE}} + L_{\text{Diff}} + L_{\text{TPR}}$
  For **SpeeDiff (w/ REPA++)**
  $L = L_{\text{VAE}} + L_{\text{Diff}} + L_{\text{TPR}} + L_{\text{REPA++}}$

  Update $(\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\theta}) \leftarrow \text{optimizerstepon}\nabla L$
end while

A.2.1. Pipeline

We summarize the training algorithm of SpeeDiff with pseudo-code in Algorithm 1. As visualized in Fig. 2, the algorithm consists of four loss branches: the VAE reconstruction loss (Appendix A.2.2), the diffusion flow matching loss (Appendix A.2.3), the Tweedie pixel reconstruction (TPR) loss (Appendix A.2.4), and the REPA++ alignment loss (Appendix A.2.5).

A.2.2. VAE Reconstruction

Given an image $\mathbf{x}_0$, the VAE encoder produces a Gaussian posterior

$$p_{\boldsymbol{\phi}}(\mathbf{z}_0 \mid \mathbf{x}_0) = \mathcal{N}\big(\mathbf{z}_0; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}_0), \sigma^2_{\boldsymbol{\phi}}(\mathbf{x}_0)\boldsymbol{I}\big), \tag{6}$$

where $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}_0)$ and $\sigma_{\boldsymbol{\phi}}(\mathbf{x}_0)$ have the same dimensionality as the latent representation. The latent sample is obtained via the reparameterization trick:

$$\mathbf{z}_0 = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}_0) + \sigma_{\boldsymbol{\phi}}(\mathbf{x}_0)\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}). \tag{7}$$

Although our VAE is not used solely for generative sampling as in classical VAEs, we find that retaining the variational formulation surprisingly benefits downstream diffusion training. This observation is further discussed in Appendix B.3. The reconstruction $\hat{\mathbf{x}}_{\text{vae}} = D_{\boldsymbol{\psi}}(\mathbf{z}_0)$ is supervised by a combination of four losses: MSE, LPIPS, discriminator loss, and KL divergence.

MSE Loss. We compute the pixel-wise mean squared error between the clean and reconstructed images. Empirically, MSE and MAE yield similar visual quality, and we adopt MSE by default.

LPIPS Loss. We use the official LPIPS implementation [62] with a VGG backbone [51]. LPIPS is computed at the original image resolution without downsampling.

Discriminator Loss. We incorporate a PatchGAN-style discriminator to enhance perceptual fidelity. To ensure stable adversarial training without relying on warm-up schedules or dynamically adjusted weights, we introduce two modifications:

- Replace BatchNorm with SpectralNorm [38] to improve Lipschitz continuity and reduce training instability.
- Apply LeCam regularization [53], which tracks an exponential moving average of real and fake logits and penalizes large deviations, effectively smoothing discriminator updates.

These stabilizations allow us to adopt a fixed discriminator loss weight and enable adversarial supervision from the beginning of training without the need for special scheduling.

Variational Objective (KL Loss). The evidence lower bound (ELBO) [28] has an additional KL divergence loss between the posterior $p_{\boldsymbol{\phi}}(\mathbf{z}_0|\mathbf{x}_0)$ and the standard Gaussian prior $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Interestingly, the KL weight strongly affects the trade-off between reconstruction of VAE and generative of diffusion. As the KL weight increases, reconstruction performance decreases while generative quality initially improves and then degrades. We select the turning point that yields the best generation performance; details are provided in Appendix B.3.

A.2.3. Diffusion Flow Matching

We adopt the linear flow-matching scheduler among several alternatives (variance preserving [22], variance exploding [52], TrigFlow [34], etc.) due to its simplicity and strong empirical performance. The stochastic interpolant is defined as $\mathbf{z}_t = (1-t)\mathbf{z}_0 + t\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. We use the velocity parameterization for diffusion model $V_{\boldsymbol{\theta}}$, which predicts the vector field of the probability flow ODE. The flow-matching loss is

$$\mathcal{L}_{\text{Diff}}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \mathbb{E}_{\mathbf{z}_0, \boldsymbol{\epsilon}, t}\left[\|V_{\boldsymbol{\theta}}(\mathbf{z}_t, t) - (\boldsymbol{\epsilon} - \mathbf{z}_0)\|_2^2\right], \tag{8}$$

where the time sampler for t is specified in Tab. 14. We observe that log-normal time sampling slightly accelerates Base model training, though the speedup becomes neglegible for Large and XL models.

A.2.4. Tweedie Pixel Reconstruction (TPR) Loss

For a given $\mathbf{x}_0$, sampled latent $\mathbf{z}_0$, and interpolant $\mathbf{z}_t$, we estimate the clean latent using Tweedie's formula:

$$\mathbf{z}_{\text{tweedie}} = \mathbf{z}_t - t\, V_{\boldsymbol{\theta}}(\mathbf{z}_t, t). \tag{9}$$

By definition, $\mathbf{z}_{\text{tweedie}} = \mathbb{E}[\mathbf{z}_0 \mid \mathbf{z}_t]$. Decoding this latent via the VAE decoder yields

$$\hat{\mathbf{x}}_{\text{tpr}} = D_{\boldsymbol{\psi}}(\mathbf{z}_{\text{tweedie}}). \tag{10}$$

This provides an additional pixel-space anchor for the diffusion model. We apply a combination of MSE and LPIPS losses between $\hat{\mathbf{x}}_{\text{tpr}}$ and $\mathbf{x}_0$.

A.2.5. REPA++

We adopt the REPA++ to align both the latent $\mathbf{z}_0$ and intermediate diffusion features $\mathbf{f}_t$ extracted from $\mathbf{z}_t$. The adapter is implemented as a two-layer MLP with SiLU activation.

A.2.6. Loss Coefficients

SpeeDiff consists of multiple loss components, yet we observe that the model is robust to a wide range of loss weights. Therefore, we adopt simple defaults: all losses use a weight of 1.0 except the discriminator loss and KL loss, which are set to 0.1 and 1×10^{-5}, respectively. The discriminator weight improves training stability, while the KL weight is selected based on the analysis in Appendix B.3.

A.3. Sampling Details

In this section, we describe our guidance sampling strategy based on AutoGuidance [25] and discuss how representation alignment affects Classifier-Free Guidance (CFG) [21] in SpeeDiff. In particular, we analyze why the gains from CFG tend to saturate when REPA++ is enabled.

A.3.1. AutoGuidance

For all SpeeDiff variants, we train an additional *bad* version of the diffusion model required by AutoGuidance. Concretely, this bad model is architecturally smaller: it uses half the number of transformer blocks and half the number of attention heads compared to the main model. To decouple it from end-to-end training, we pair this bad diffusion model with the frozen VAE obtained from the 200-epoch checkpoint and train the bad diffusion model from scratch for only 20 epochs.

At sampling time, we apply AutoGuidance using this bad model in conjunction with the main model. The guidance scale and guidance intervals are tuned per model size to achieve the best gFID score.

A.3.2. Analysis of Unconditional Generation and CFG

We compare three settings for guidance: (1) without guidance, (2) CFG, and (3) AutoGuidance (AG). For SpeeDiff (w/ REPA++), we observe that the improvement from CFG over the unguided model is noticeably saturated, whereas AG continues to provide a substantial gain.

After careful analysis, we identify a key contributing factor: the unconditional generation of SpeeDiff (with REPA++) is already surprisingly strong. In other words, the null-class (unconditional) predictions of the main model produce high-quality samples. We illustrate unconditional samples in Fig. 6 and report their FID scores in Tab. 8. Due to the strong unconditional sampling quality, the negative direction in CFG provides only limited additional guidance. AutoGuidance avoids this issue by constructing the negative direction using the prediction of a deliberately degraded bad model under the conditional class. This alternative direction is substantially more informative, and as a result, AutoGuidance delivers a much larger improvement in FID compared to CFG.

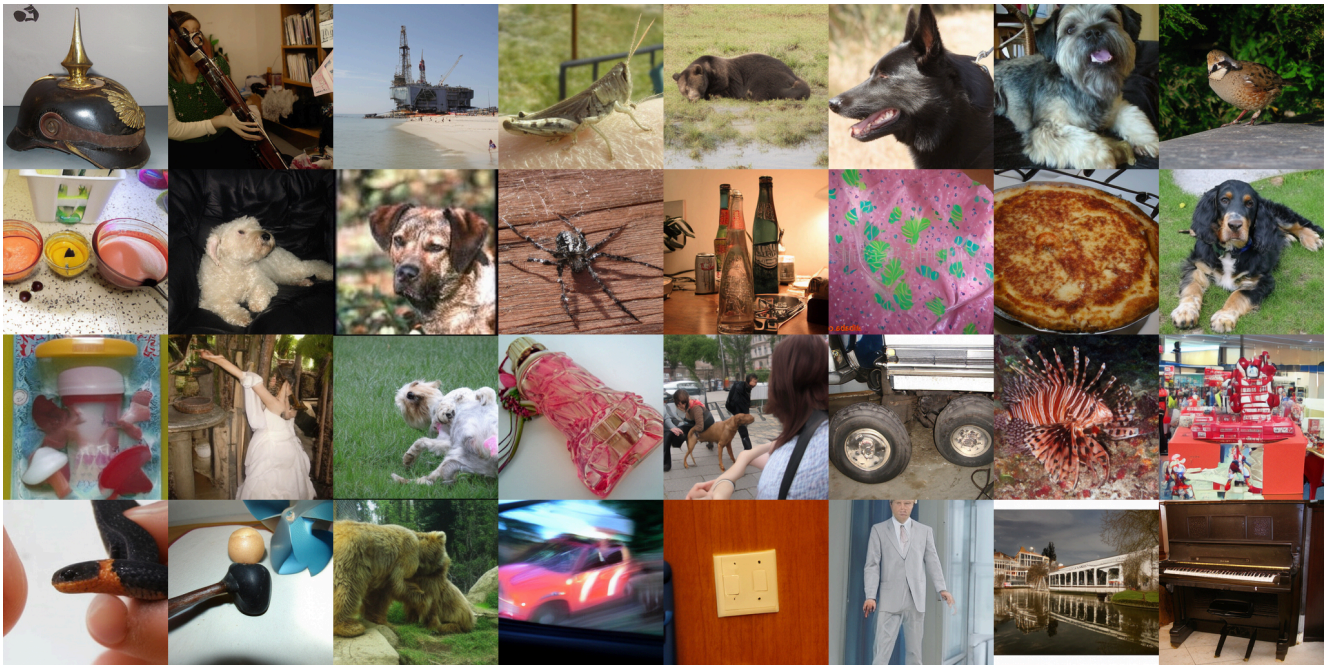


Figure 6. **Uncurated samples of unconditional generation.** On ImageNet 512×512 with our SpeeDiff-XL (w/ REPA++) model (200 epochs) with the null-class. The unconditional generation is already surprisingly strong.

Table 8. **Unconditional Generation.** gFID on ImageNet 256×256 after 200 epochs.

Method	gFID↓ (unconditional)
SpeeDiff-XL (w/o REPA++)	22.32
SpeeDiff-XL (w/ REPA++)	4.03

Table 9. **CFG vs. AG.** gFID on ImageNet 256×256 after 200 epochs. CFG yields limited gains for SpeeDiff (w/ REPA++) due to its strong unconditional generation.

Method	gFID↓ (w/o guidance)	gFID↓ (w/ CFG)	gFID↓ (w/ AG)
SpeeDiff-XL (w/o REPA++)	2.42	1.52	1.46
SpeeDiff-XL (w/ REPA++)	1.50	1.45	1.21

Table 10. **ImageNet 256×256 generation benchmark (continue).** We report additional results for using different VAE and diffusion architectures to demonstrate the superior performance of refined architecture for both non–representation alignment and representation alignment methods (↓ lower is better; ↑ higher is better).

Method	VAE		Diffusion		Epochs	Generation w/o guidance					Generation w/ guidance					rFID↓
	Model	Params (M)	Model	Params (M)		gFID↓	sFID↓	IS↑	Prec.↑	Rec.↑	gFID↓	sFID↓	IS↑	Prec.↑	Rec.↑	
Non-representation Alignment Methods																
SpeeDiff (w/o REPA++)	CNN-VAE	70	DiT-XL/1	675	80	5.89	5.49	145.58	0.75	0.61	2.25	5.35	254.22	0.81	0.57	0.73
	CNN-VAE	70	Refined-DiT-B	91	80	9.24	5.53	111.26	0.72	0.59	3.50	5.23	200.70	0.80	0.54	0.69
			Refined-DiT-L	319	80	4.62	4.53	150.40	0.77	0.59	2.17	4.75	251.28	0.82	0.56	0.67
			Refined-DiT-XL	672	80	3.71	4.53	166.52	0.78	0.61	2.12	4.69	252.60	0.83	0.56	0.61
	ViT-VAE-B	58	Refined-DiT-B	91	80	10.65	5.67	105.59	0.71	0.59	3.86	5.13	198.07	0.81	0.54	0.74
	ViT-VAE-L	207	Refined-DiT-L	319	80	5.31	4.78	144.62	0.76	0.61	2.15	4.65	245.47	0.82	0.58	0.58
	ViT-VAE-XL	437	Refined-DiT-XL	672	80	3.66	4.42	172.03	0.78	0.62	1.80	4.40	267.10	0.81	0.59	0.54
Representation Alignment Methods																
SpeeDiff (w/ REPA++)	CNN-VAE	70	DiT-XL/1	675	80	2.83	4.31	185.36	0.78	0.62	1.80	4.37	273.49	0.79	0.63	0.62
	CNN-VAE	70	Refined-DiT-B	91	80	5.82	5.08	142.07	0.75	0.60	2.78	5.28	216.66	0.78	0.58	0.62
			Refined-DiT-L	319	80	2.65	4.33	183.23	0.79	0.61	1.75	4.66	271.58	0.80	0.61	0.60
			Refined-DiT-XL	672	80	2.15	4.21	206.49	0.80	0.62	1.74	4.55	279.13	0.81	0.61	0.64
	ViT-VAE-B	58	Refined-DiT-B	91	80	6.20	5.12	138.71	0.74	0.60	2.29	4.83	230.33	0.79	0.57	0.84
	ViT-VAE-L	207	Refined-DiT-L	319	80	2.15	4.21	207.87	0.79	0.61	1.41	4.15	292.90	0.80	0.61	0.62
	ViT-VAE-XL	437	Refined-DiT-XL	672	80	1.69	4.14	234.56	0.80	0.63	1.29	4.08	309.42	0.80	0.63	0.59

B. Additional Results and Ablations

B.1. Joint Scaling of VAE and Diffusion

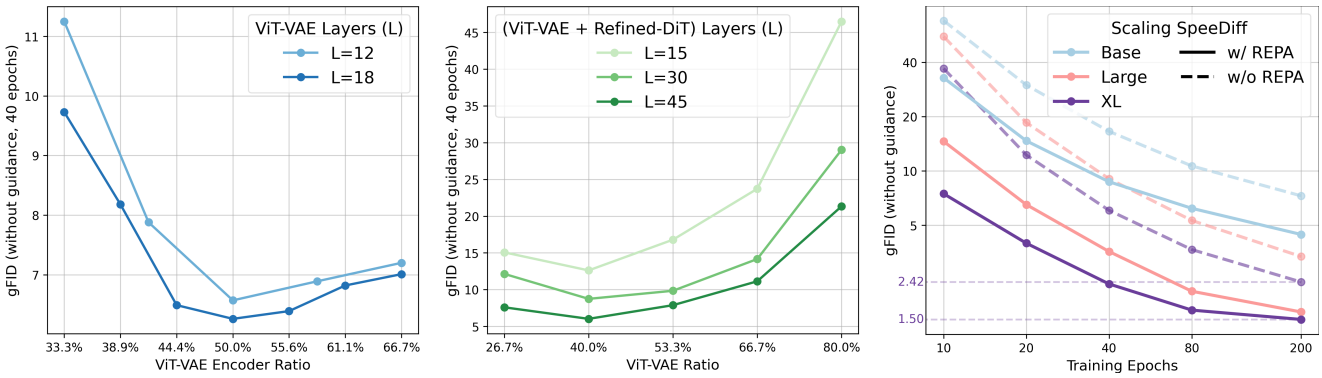
In this section, we examine how jointly scaling the VAE and the diffusion model affects performance. Unlike prior latent-diffusion pipelines that pair a fixed-size VAE with diffusion models of various capacities, SpeeDiff benefits substantially from joint scaling both components. A larger VAE not only produces a more expressive latent space but also provides stronger feedback to the diffusion model during end-to-end training.

To understand how model capacity should be allocated, we systematically vary the number of transformer layers assigned to the encoder, decoder, and diffusion model. As shown in Fig. 7, several trends emerge. First, the encoder and decoder perform best when their depths are approximately balanced (around a 1:1 ratio). Second, dedicating roughly 40% of total layers to the VAE yields a strong trade-off between reconstruction quality and downstream generative performance. Finally, we observe a consistent and stable scaling trend across model sizes, both with and without REPA++, when following a 1:1:3 allocation rule for encoder, decoder, and diffusion layers.

B.2. SpeeDiff with Conventional Architecture

We further evaluate the impact of our refined architectures by comparing ViT-VAE and Refined-DiT with conventional CNN-VAE and DiT variants. As shown in Tab. 10, both refinements consistently improve performance across model scales.

Across both non-aligned and REPA++ settings, our refined architectures provide consistent improvements over conventional CNN-VAE and DiT designs. Without alignment, replacing DiT with Refined-DiT already yields substantial gains (e.g., gFID improves from 5.89 to 3.71 with CNN-VAE), and replacing CNN-VAE with ViT-VAE further enhances performance, reaching 3.66 at the XL scale on ImageNet 256×256 generation after 80 epochs. With REPA++, the benefits become even more pronounced: Refined-DiT-XL lowers gFID from 2.83 to 2.15, and jointly scaling the VAE achieves the best overall result of 1.69 on ImageNet 256×256 generation after 80 epochs. Together, these findings confirm the effectiveness of our refined architectures.



a. VAE encoder and decoder **b**. VAE and diffusion **c**. Joint scaling

Figure 7. **Joint scaling of VAE and diffusion.** On ImageNet 256×256 generation, we examine how to allocate transformer layers between encoder, decoder, and diffusion. (a) Encoder and decoder perform best with a balanced 1:1 ratio. (b) Allocating about 40% of total layers to the VAE gives a strong trade-off. (c) A 1:1:3 allocation (encoder:decoder:diffusion) consistently works well with and without REPA++.

B.3. Analysis the Effect of KL Loss

The variational objective (KL loss) in the VAE loss plays a crucial role in balancing reconstruction of VAE and generative of diffusion. Recall that the KL term measures the divergence between the encoder posterior $p_{\boldsymbol{\phi}}(\mathbf{z}_0 \mid \mathbf{x}_0)$ and the standard Gaussian prior $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$:

$$D_{\mathrm{KL}}\big(\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\phi}}, \sigma_{\boldsymbol{\phi}}^2 \boldsymbol{I}) \,\|\, \mathcal{N}(\mathbf{0}, \boldsymbol{I})\big) = \frac{1}{2}\Bigg[\underbrace{\sum_{i=1}^{d} \boldsymbol{\mu}_{\boldsymbol{\phi},i}^2}_{\text{mean penalty}} + \underbrace{\sum_{i=1}^{d} \left(\sigma_{\boldsymbol{\phi},i}^2 - \log \sigma_{\boldsymbol{\phi},i}^2 - 1\right)}_{\text{variance penalty}} \Bigg]. \tag{11}$$

Through extensive experiments, we find that the weight assigned to this KL term has a surprisingly strong influence on downstream diffusion performance. Setting the KL weight to zero effectively turns the model into a standard autoencoder (AE). Although this preserves reconstruction quality, it consistently harms generative performance, suggesting that a weakly regularized latent space is less suitable for diffusion training. Conversely, assigning too large a KL weight forces the latent distribution toward an overly restrictive prior, degrading both reconstruction and generation. Empirically, we observe that the average predicted standard deviation $\sigma_{\boldsymbol{\phi},i}$ increases monotonically as the KL weight is raised from zero to sufficiently large values. This trend is summarized in Tab. 11, based on which we select the turning point of the KL weight that provides the best generative performance.

Table 11. **Effect of KL Loss.** We study how the KL loss shapes the latent space by measuring the average latent log-variance $\log \sigma_{\boldsymbol{\phi}}^2$ across spatial locations and samples. We additionally report both generation and reconstruction metrics on ImageNet 256×256 after 80 epochs with SpeeDiff-XL (w/ REPA++).

KL weight	0 (AE)	1e-6	3e-6	1e-5 (*best*)	3e-5	1e-4
Average log-variance	-123.31	-13.26	-9.25	-6.87	-4.31	-2.13
gFID↓	3.91	2.48	1.94	**1.69**	1.86	2.76
rFID↓	**0.34**	0.39	0.44	0.59	0.87	1.46
PSNR↑	**27.59**	27.33	26.98	26.05	24.83	22.08

B.4. Sensitivity to TPR Loss Weight

We ablate the TPR loss weight to examine the sensitivity of SpeeDiff to this hyperparameter. As shown in Tab. 12, performance remains stable across a wide range of weights (0.5–4.0), with all settings yielding comparable gFID, rFID, and PSNR. Notably, even a $4\times$ increase from the default weight of 1.0 causes negligible degradation in generation quality (gFID 6.33 vs. 6.20), while slightly improving reconstruction metrics. This robustness indicates that SpeeDiff does not require careful tuning of the TPR loss weight in practice.

Table 12. Ablation on TPR loss weight with SpeeDiff-B (w/ REPA++) on ImageNet 256 × 256 after 80 epochs.

TPR weight	**0.0**	**0.5**	**1.0 (default)**	**2.0**	**4.0**
gFID↓	12.69	6.91	6.20	**6.08**	6.33
rFID↓	0.90	0.85	0.84	0.80	**0.74**
PSNR↑	25.40	25.56	25.61	25.65	**25.87**

B.5. Analysis of Practical Cost

We analyze the practical cost of SpeeDiff by measuring the time-level speedup attributable to end-to-end training. Specifically, we compare SpeeDiff (w/o REPA++) against the two-stage baseline, which assumes a pretrained VAE with preprocessed latents and thus requires only *a single diffusion forward pass per training step*. In contrast, our end-to-end approach additionally involves *one VAE encoder call and two decoder calls per step*, resulting in an approximately 2× slower per-step time (see Tab. 13 for detailed per-step timing and Tab. 7 for model FLOPs).

Despite this per-step overhead, SpeeDiff converges dramatically faster in terms of training steps. Taking the baseline's 7.93 FID without guidance at 4M steps as the reference point, SpeeDiff reaches the same FID in only 0.17M steps. This translates to a 23.52× step-level speedup and, after accounting for the per-step cost difference, a 12.14× time-level speedup. Notably, these numbers are conservative: the two-stage pipeline additionally requires VAE pretraining or finetuning as a prerequisite, whereas SpeeDiff trains the VAE jointly from scratch, further widening the practical efficiency gap.

Table 13. **Practical cost and speedup**. We evaluate on ImageNet 256×256 with 16 H100-SXM-80GB GPUs at batch size 256.

	Step time (s)	Required steps (M)	Step-level speedup	Time-level speedup
Two-stage-XL	0.16	4	–	–
SpeeDiff-XL	0.31	0.17	23.52×	12.14×

Regarding memory consumption, SpeeDiff-XL requires 57.34 GB peak memory compared to 26.97 GB for the two-stage baseline under the same training configuration. While the higher memory footprint is a limitation of end-to-end training, we consider it a reasonable trade-off given the substantial convergence speedup achieved.

C. Additional Qualitative Samples

We show additional samples of SpeeDiff-XL on both ImageNet 256×256 and 512×512 generation in Figs. 8 to 12 and Figs. 13 to 17, respectively.

Table 14. **Detailed implementation of SpeeDiff on ImageNet at 256×256 and 512×512 resolutions.** We report architecture, loss functions, training, and sampling hyperparameters. Rows highlighted in gray indicate values that differ across models.

	SpeeDiff-B (256)	**SpeeDiff-L** (256)	**SpeeDiff-XL** (256)	**SpeeDiff-B** (512)	**SpeeDiff-L** (512)	**SpeeDiff-XL** (512)
ViT-VAE details						
Size	Base	Large	XL	Base	Large	XL
Patch size	16	16	16	32	32	32
Layers	4	8	9	4	8	9
Hidden dims	768	1024	1408	768	1024	1408
Numbers of Heads	12	16	22	12	16	22
Dims per heads	64	64	64	64	64	64
Latent channels	32	32	32	32	32	32
Refined-DiT details						
Size	Base	Large	XL	Base	Large	XL
Layers	12	24	27	12	24	27
Hidden dims	768	1024	1408	768	1024	1408
Numbers of Heads	12	16	22	12	16	22
Dims per heads	64	64	64	64	64	64
Patch size	1	1	1	1	1	1
Class drop ratio	10%	10%	10%	10%	10%	10%
VAE loss details						
MSE weight	1.0	1.0	1.0	1.0	1.0	1.0
LPIPS weight	1.0	1.0	1.0	1.0	1.0	1.0
GAN weight	0.1	0.1	0.1	0.1	0.1	0.1
GAN starting step	0	0	0	0	0	0
GAN LeCam weight	0.05	0.05	0.05	0.05	0.05	0.05
GAN LeCam EMA	0.999	0.999	0.999	0.999	0.999	0.999
KL weight	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5
Diffusion loss details						
Time sampler	Log-norm (0.0, 1.0)	Uniform (0.0, 1.0)	Uniform (0.0, 1.0)	Log-norm (0.0, 1.0)	Uniform (0.0, 1.0)	Uniform (0.0, 1.0)
Loss weight	1.0	1.0	1.0	1.0	1.0	1.0
Scheduler	Linear	Linear	Linear	Linear	Linear	Linear
Tweedie Pixel Reconstruction (TPR) loss details						
L2 weight	1.0	1.0	1.0	1.0	1.0	1.0
LPIPS weight	1.0	1.0	1.0	1.0	1.0	1.0
REPA++ loss details						
Latent REPA weight	1.0	1.0	1.0	1.0	1.0	1.0
Diffusion REPA weight	1.0	1.0	1.0	1.0	1.0	1.0
Diffusion REPA depth	8	8	8	8	8	8
Data augmentation details						
Center Crop	✓	✓	✓	✓	✓	✓
Random Horizontal Flip	✓	✓	✓	✓	✓	✓
Training details						
Batch size	256	256	256	256	256	256
Iterations	1M	1M	1M	1M	1M	1M
Epochs	200	200	200	200	200	200
Learning rate	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4
Adam β_1	0.9	0.9	0.9	0.9	0.9	0.9
Adam β_2	0.999	0.999	0.999	0.999	0.999	0.999
Weight decay	0.0	0.0	0.0	0.0	0.0	0.0
EMA decay rate	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
Sampling details						
SDE	✓	✓	✓	✓	✓	✓
Solver	Euler–Maruyama	Euler–Maruyama	Euler–Maruyama	Euler–Maruyama	Euler–Maruyama	Euler–Maruyama
NFE	250	250	250	250	250	250

Figure 8. **Uncurated samples without guidance.** On ImageNet 256×256 with our SpeeDiff-XL (w/ REPA++) model (200 epochs). The class labels are *loggerhead* (33, left) and *macaw* (88, right).

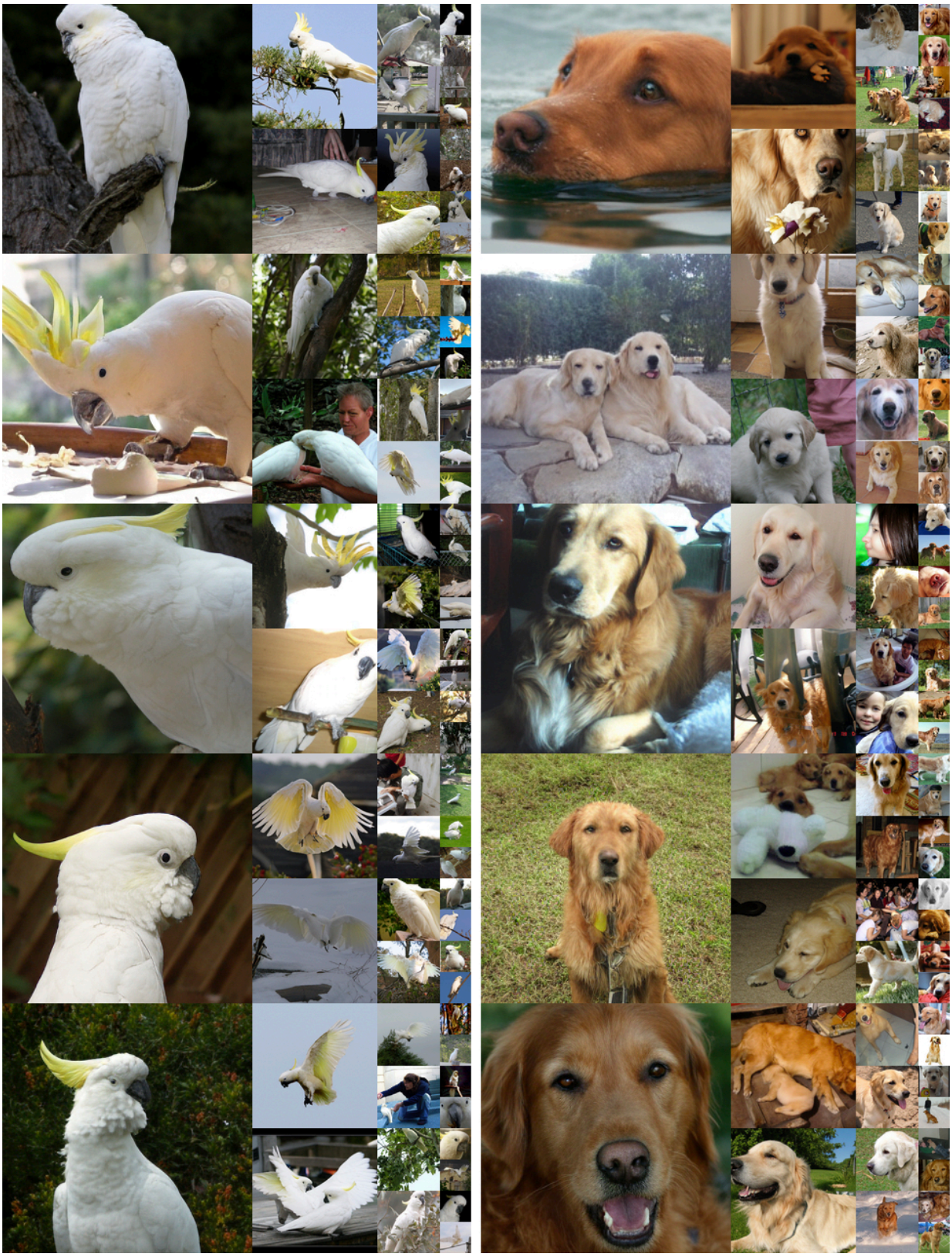


Figure 9. **Uncurated samples without guidance.** On ImageNet 256×256 with our SpeeDiff-XL (w/ REPA++) model (200 epochs). The class labels are *sulphur-crested cockatoo* (89, left) and *golden retriever* (207, right).

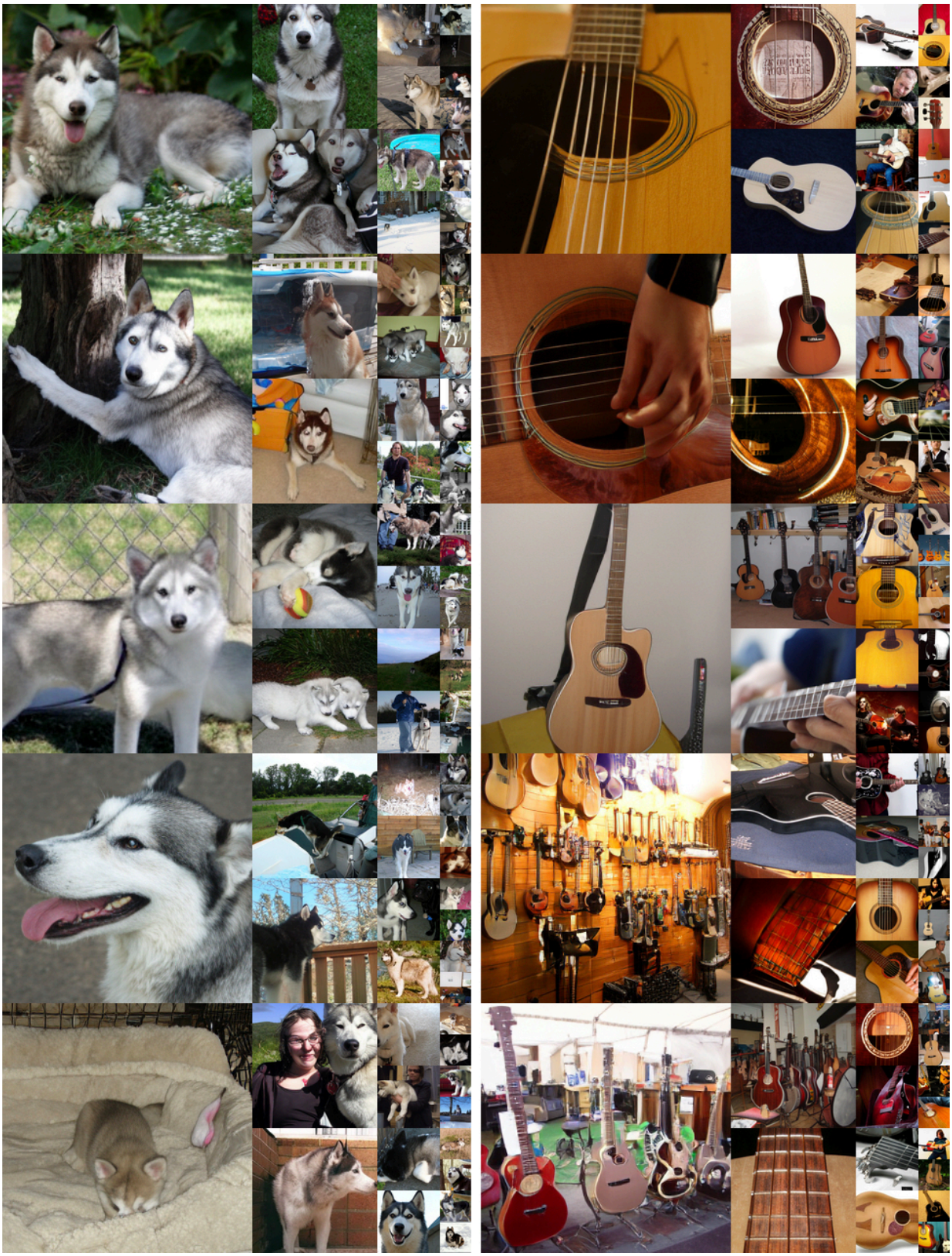


Figure 10. **Uncurated samples without guidance.** On ImageNet 256×256 with our SpeeDiff-XL (w/ REPA++) model (200 epochs). The class labels are *Siberian husky* (250, left) and *acoustic guitar* (402, right).

Figure 11. **Uncurated samples without guidance.** On ImageNet 256×256 with our SpeeDiff-XL (w/ REPA++) model (200 epochs). The class labels are *baseball* (429, left) and *dogsled* (537, right).

Figure 12. **Uncurated samples without guidance.** On ImageNet 256×256 with our SpeeDiff-XL (w/ REPA++) model (200 epochs). The class labels are *fire truck* (555, left) and *volcano* (980, right).

Figure 13. **Uncurated samples without guidance.** On ImageNet 512×512 with our SpeeDiff-XL (w/ REPA++) model (200 epochs). The class labels are *white wolf* (270, left) and *Persian cat* (283, right).

Figure 14. **Uncurated samples without guidance.** On ImageNet 512×512 with our SpeeDiff-XL (w/ REPA++) model (200 epochs). The class labels are *lion* (291, left) and *otter* (360, right).

Figure 15. **Uncurated samples without guidance.** On ImageNet 512×512 with our SpeeDiff-XL (w/ REPA++) model (200 epochs). The class labels are *African elephant* (386, left) and *panda* (388, right).

Figure 16. **Uncurated samples without guidance.** On ImageNet 512×512 with our SpeeDiff-XL (w/ REPA++) model (200 epochs). The class labels are *castle* (483, left) and *sports car* (817).

Figure 17. **Uncurated samples without guidance.** On ImageNet 512×512 with our SpeeDiff-XL (w/ REPA++) model (200 epochs). The class labels are *suspension bridge* (839, left) and *pizza* (963).