SimFlow: Simplified and End-to-End Training of Latent Normalizing Flows

Qinyu Zhao[1,2], Guangting Zheng[2], Tao Yang[2], Rui Zhu[2,†], Xingjian Leng[1], Stephen Gould[1], and Liang Zheng[1]

[1] Australian National University, Canberra ACT, Australia
[2] ByteDance Seed, Beijing, China
† Project lead
Project: https://qinyu-allen-zhao.github.io/SimFlow/
Contact: qinyu.zhao@anu.edu.au

Abstract. Normalizing Flows (NFs) learn invertible mappings between the data and a Gaussian distribution. Prior works usually suffer from two limitations. First, they add random noise to training samples or VAE latents as data augmentation, introducing complex pipelines including extra noising and denoising steps. Second, they use a pretrained and frozen VAE encoder, resulting in suboptimal reconstruction and generation quality. In this paper, we find that the two issues can be solved in a very simple way: just fixing the variance (which would otherwise be predicted by the VAE encoder) to a constant (e.g., 0.5). On the one hand, this method allows the encoder to output a broader distribution of tokens and the decoder to learn to reconstruct clean images from the augmented token distribution, avoiding additional noise or denoising design. On the other hand, fixed variance simplifies the VAE evidence lower bound, making it stable to train an NF with a VAE jointly. On the ImageNet 256×256 generation task, our model SimFlow obtains a gFID score of 2.15, outperforming the state-of-the-art method STARFlow (gFID 2.40). Moreover, SimFlow can be seamlessly integrated with the end-to-end representation alignment (REPA-E) method and achieves an improved gFID of 1.91, setting a new state of the art among NFs.

1 Introduction

Normalizing flows (NFs) [10, 16, 17, 32, 51, 70, 72] model a data distribution by transforming a prior distribution (*e.g.*, the normal distribution) via invertible mappings. For easier training and better generation, state-of-the-art methods [16, 17] adopt two strategies: 1) using a variational autoencoder (VAE) [52] to train a latent NF, and 2) adding noise to VAE latents during NF training as data augmentation.

However, there are two limitations with the above strategies. First, while adding random noise smooths latent space for more generalizable NF modeling, it complicates the pipelines. Specifically, NFs trained with noisy inputs generate noisy outputs, so existing works have to introduce an additional denoising

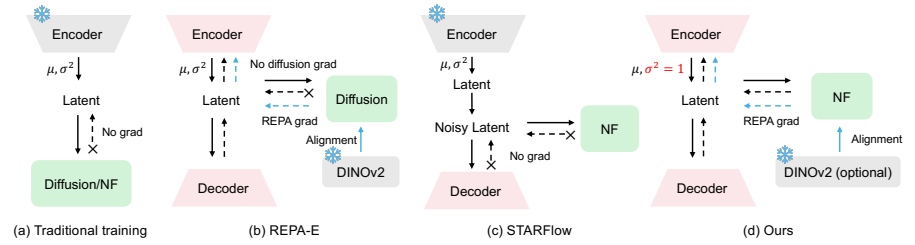

Fig. 1: Comparison of our framework with closely-related methods. (a) Standard practice trains a VAE first and then train a generative model with the VAE frozen. Note that, for each image, the VAE encoder outputs the mean $\boldsymbol{\mu}$ and the variance $\boldsymbol{\sigma}^2$ of a Gaussian, from which a set of tokens is sampled. The variance $\boldsymbol{\sigma}^2$ is usually very small in a standard pretrained VAE. (b) REPA-E [36] jointly trains diffusion with VAE using the REPA loss [69], and the diffusion gradient is stopped before the VAE to avoid latent collapse. (c) STARFlow [16] trains NF and decoder on noisy latent with the encoder frozen. (d) We train an NF and a VAE in an end-to-end way from scratch. There is no stop-gradient operator, significantly simplifying prior frameworks. Solid arrows indicate the forward pass, while dashed arrows denote gradient flows. We label frozen modules in gray, generative models in green, and trainable VAE modules in red.

process, such as score-based denoising [70], a fine-tuned VAE decoder [16], or a flow matching model for denoising [17]. The extra noising and denoising steps complicate the training and inference processes.

Second and more importantly, these methods use a frozen VAE encoder, which leads to suboptimal reconstruction and generation quality. For *reconstruction*, the encoder is sensitive to noise because it is not trained under noise augmentation introduced for NFs. Possibly, if images are very close to each other in the latent space, noise perturbations may severely degrade the latents. While it is feasible to fine-tune the decoder [16], reconstruction quality remains poor, because some encoded information is already lost due to noise (for results see Section 5.2). For *generation*, now that the encoder is not optimized jointly with an NF, the latent space may not be best suitable for NF modeling.

In this paper, we solve both issues at the same time by introducing SimFlow, a simple end-to-end training framework for latent NFs. Our key idea is to fix the encoder-predicted variance to a constant (for instance, 0.5). On the one hand, this method has similar effects with adding noise to latents [16], that is, to smooth the latent space and improve generalization, but greatly simplifies the pipeline. Note that pretrained VAEs tend to predict extremely small variance (*e.g.*, less than 0.01), which collapses a token distribution to nearly a single point. In contrast, our method of fixing variance to a larger value ensures each latent token is sampled from a broader distribution rather than a nearly deterministic result. Meanwhile, the decoder learns how to reconstruct clean images from the augmented token distributions during VAE training. In this way, our approach removes the need for additional noising or denoising design.

On the other hand, fixing the variance to a value like 0.5 enables effective end-to-end training of latent NFs, making the VAE latents compatible with NF modeling. From the VAE evidence lower bound (ELBO) perspective, this method makes the entropy term become a constant, which simplifies the training objective to consist of only the reconstruction loss and the generation loss. We find that it is much easier to balance between reconstruction and generation than to train the whole framework. A comparison of our framework with prior works is shown in Figure 1.

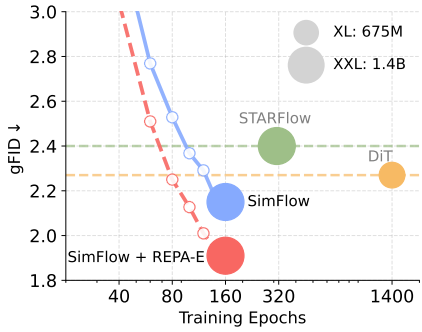

Fig. 2: Comparing SimFlow with prior works. On ImageNet 256 × 256, our model SimFlow, with fewer training epochs, achieves better generation quality than the prior best NF-based model STARFlow [16].

On the ImageNet 256 × 256 generation benchmark, SimFlow achieves a gFID score of 2.15, surpassing the prior work STARFlow (gFID 2.40) [16], as shown in Figure 2. This is also the first report that NFs outperform DiT [47], a representative diffusion model. While SimFlow has a larger model size, its convergence rate is more than 8 times faster than DiT. Importantly, our framework is fully compatible with REPA-E [36]: aligning NF features with DINOv2 [4] and training both the VAE and NF using the alignment loss. Combining SimFlow with REPA-E further decreases gFID to 1.91 on Image 256 × 256 and achieves a gFID score of 2.74 on ImageNet 512 × 512, establishing new state-of-the-art performance among NF-based models.

2 Related work

Normalizing flows (NFs) [9–12, 14, 32, 34, 40, 41, 44, 51, 62, 72] are a useful framework for density estimation, visual generation, and text generation. In this paper, we mainly adopt autoregressive flows (AFs) [16, 70]. In each invertible transformation of an AF, each token is transformed conditioned on previous tokens. Representative AFs include IAF [33], MAF [45], neural AF [24], and T-NAF [46]. More recently, TARFlow [70] leverages causal Transformers and simplifies the log-determinant term in the loss function, leading to notable improvements in generation quality. STARFlow [16] extends TARFlow into the VAE latent space and further improves the NF performance.

VAEs with fixed variance. A prior work [2] fix the variance of the VAE encoder for structured sequence prediction. Sun et al. [58] introduce σ-VAE for autoregressive generation, which has a fixed variance, and later works adopted this technique [18, 29]. Although these methods and SimFlow both fix the variance, they differ in several ways. **Motivation.** σ-VAE fixes the variance mainly to mitigate accumulated errors in AR generation. In contrast, we derive the fixed variance from the VAE+NF ELBO to stabilize end-to-end training and prevent

the latent collapse problem. **Other contributions.** We further provide a theoretical analysis of why fixed variance avoids collapse in Sec. 4.3, and empirically validate its effect in Sec. 4.4. In Section 4.4, we show that our perspective provides a unified explanation of why various perturbation methods in latent space actually allow for stable end-to-end training.

Joint training of generative models with VAEs. The standard practice is to train a VAE on the reconstruction task first and then train a generative model with the VAE frozen [1, 16, 37, 39, 43, 47, 49, 50, 57]. But an important question is whether the pretrained VAE space is suitable for training a generative model. Without solving this, a possible consequence is that a VAE with excellent reconstruction ability leads to poor generation quality [35, 67].

End-to-end training of generative models with VAEs is appealing because it not only streamlines the training pipeline but is also expected to make the VAE latent space more suitable for generation. Early exploration [51, 56, 61] is not stable in training and does not achieve competitive performance on a large-scale dataset like ImageNet 256×256. Recently, REPA-E [36] reported an FID of 1.12 on ImageNet 256×256 with end-to-end training. In REPA-E, the gradient of the REPA loss [69] flows through the diffusion model to the VAE, while the gradient from the diffusion model does not flow to the VAE (see Figure 1(b)). When the latter is allowed, they observe latent space collapse, where the latent variance shrinks quickly, and the generation quality is low. Different from REPA-E, SimFlow jointly trains NFs and VAEs from scratch without stopping any gradient, further simplifying the training framework.

3 Preliminaries

A standard VAE [31] consist of an encoder $q_{\boldsymbol{\psi}}$ and a decoder $p_{\boldsymbol{\omega}}$. Given an image $\mathbf{i}$, the encoder predicts mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$ of a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$. A latent variable $\mathbf{x}$ (a set of tokens) is sampled from $\mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$. The decoder is trained to reconstruct the image $\mathbf{i}$ from $\mathbf{x}$. The VAE is trained with ELBO as follows:

$$\log p(\mathbf{i}) \geq \mathbb{E}_{\mathbf{x}\sim q_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{i})}\left[\underbrace{\log p_{\boldsymbol{\omega}}(\mathbf{i} \mid \mathbf{x})}_{\text{Reconstruction}} + \underbrace{\log p(\mathbf{x})}_{\text{Prior}} - \underbrace{\log q_{\boldsymbol{\psi}}(\mathbf{x} \mid \mathbf{i})}_{\text{Entropy}}\right], \tag{1}$$

where the prior term is usually chosen as normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$, and prior and entropy terms are combined into a Kullback-Leibler (KL) divergence term.

Latent NFs [9, 10, 16, 32, 51] map the VAE latent distribution into a simple one $\mathbf{z} \sim p_0(\mathbf{z})$ (the normal distribution), via learning an invertible function $f_{\boldsymbol{\theta}}$. During training, the forward pass maps the sampled latent $\mathbf{x}$ to $\mathbf{z} = f_{\boldsymbol{\theta}}(\mathbf{x})$, following the change of variable formula:

$$p_{\mathrm{NF}}(\mathbf{x}; \boldsymbol{\theta}) = p_0(\mathbf{z}) \left|\det\left(\tfrac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)\right| = p_0(f_{\boldsymbol{\theta}}(\mathbf{x})) \left|\det\left(\tfrac{\partial f_{\boldsymbol{\theta}}(\mathbf{x})}{\partial \mathbf{x}}\right)\right|. \tag{2}$$

NFs are trained with maximum likelihood estimation:

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}\sim p_{\text{latent}}} \log p_{\mathrm{NF}}(\mathbf{x}; \boldsymbol{\theta}). \tag{3}$$

At inference, $\mathbf{z}$ is sampled from the target distribution $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and an inverse process maps $\mathbf{z}$ to $\mathbf{x} = f_{\boldsymbol{\theta}}^{-1}(\mathbf{z})$, which is then decoded by the decoder to images. In this way, $f_{\boldsymbol{\theta}}^{-1}$ with the VAE decoder is a generative model, which maps Gaussian noise to new image samples.

This study mainly builds on STARFlow [16], as it achieves competitive performance compared to other generative models on the large-scale ImageNet dataset. That said, our end-to-end training framework is expected to be applicable to other latent NF models as well.

4 Method

In Section 4.1, we present SimFlow, our joint training framework from two different perspectives: noise augmentation in NF training and the VAE ELBO formulation. In Section 4.2, we adopt REPA-E to further improve SimFlow. In Section 4.4, latent space is analyzed to show the effects of end-to-end training. Section 4.5 revisits and improves the classifier-free guidance (CFG) for NFs.

4.1 SimFlow: End-to-end training of latent NFs

Let $f_{\boldsymbol{\theta}}$ denote the NF model, and let $q_{\boldsymbol{\psi}}$ and $p_{\boldsymbol{\omega}}$ denote the encoder and decoder of a VAE, respectively. We set variance outputted from the encoder to constant $\bar{\sigma}^2$. The encoder only predicts mean $\boldsymbol{\mu}$ for each image $\mathbf{i}$, and the latent variable $\mathbf{x}$ is sampled from Gaussian $\mathcal{N}(\boldsymbol{\mu}, \bar{\sigma}^2\mathbf{I})$. The NF is trained to convert $\mathbf{x}$ to Gaussian samples $\mathbf{z} = f_{\boldsymbol{\theta}}(\mathbf{x})$, and the decoder learns to reconstruct the images from $\mathbf{x}$.

Our system is trained with the combined VAE and NF objective (the full derivation is provided in the supplement):

$$\max_{\boldsymbol{\theta},\boldsymbol{\psi},\boldsymbol{\omega}} \mathbb{E}_{\mathbf{i}\sim p_{\text{data}}} \mathbb{E}_{\mathbf{x}\sim q_{\boldsymbol{\psi}}(\mathbf{x}|\mathbf{i})} \left[\log p_{\boldsymbol{\omega}}(\mathbf{i} \mid \mathbf{x}) + \log p_{\text{NF}}(\mathbf{x}; \boldsymbol{\theta})\right], \tag{4}$$

where the entropy term is a constant and thus omitted. Note that we do not tune the weights of loss terms in Eq. (4) and simply set them to 1.0. We also add perceptual loss and adversarial loss [15] as common practice in VAE training [52].

From the noise-augmented training perspective, the distribution outputted by the VAE encoder can be decomposed into $\boldsymbol{\mu}$ and a Gaussian noise, $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \text{diag}(\boldsymbol{\sigma}^2))$. Ideally, the token distribution is already augmented by noise $\boldsymbol{\epsilon}$. However, for a well-pretrained VAE, we observe that $\boldsymbol{\sigma}^2$ is usually very small. There are two main reasons. First, the KL term could have maintained the variance as regularization but is usually assigned a very small weight (*e.g.*, 10^{-5}), making its effect negligible. Second, if $\boldsymbol{\sigma}^2$ is large, it is hard for the decoder to reconstruct the image from the highly varying latent. Thus, the encoder shrinks the variance, making it easier to decode. As a result, the latent still needs noise augmentation for NF training.

In this work, we manually set σ^2 from the encoder to a constant $\bar{\sigma}^2$ and the VAE encoder embeds images into $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \bar{\sigma}^2\mathbf{I})$. The resulting 'VAE+NF' framework naturally becomes noise-augmented training. NF learns

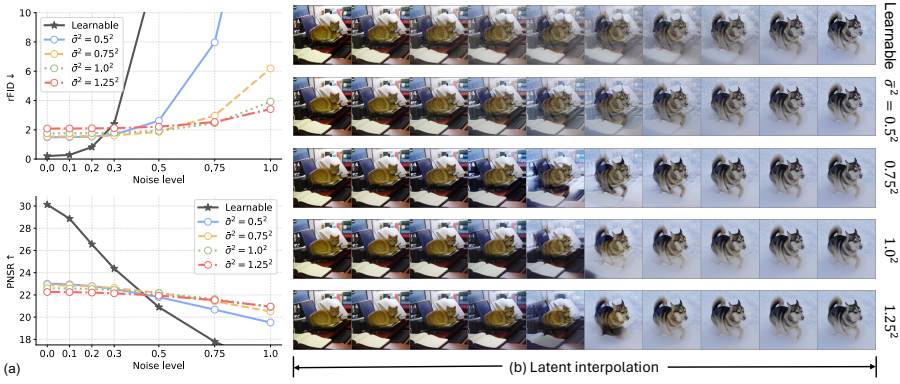

Fig. 3: Robustness of VAEs with fixed variances. (a) A VAE with a large and fixed variance can maintain reconstruction quality under latent noise, while the performance of a VAE with learnable variance degrades significantly. (b) For VAEs with a large variance, the images reconstructed from linearly interpolated latents still clearly show the main subjects (the cat or the dog), rather than blending them. 'Learnable' indicates a standard VAE with learnable variance, while '$\bar{\sigma}^2 = x^2$' denotes a VAE with a fixed variance of x^2.

the distribution while the decoder is trained to reconstruct the image. This avoids additional noising and denoising steps.

From the VAE ELBO perspective, a straightforward way for end-to-end training is to follow the ELBO:

$$\mathbb{E}_{\mathbf{x}\sim q_{\psi}(\mathbf{x}|\mathbf{i})}\left[\log p_{\boldsymbol{\omega}}(\mathbf{i}\mid\mathbf{x}) + \log p_{\mathrm{NF}}(\mathbf{x};\boldsymbol{\theta}) - \log q_{\psi}(\mathbf{x}\mid\mathbf{i})\right], \tag{5}$$

where the entropy term maintains variance not to be small, avoiding latent collapse [36, 61].

However, we empirically find it hard to balance the three loss terms. The reconstruction term tends to reduce the variance, the NF term tends to shrink the predicted latent mean, while the entropy term tends to enlarge the variance. The tension makes the training sensitive to the weighting of these terms and leads to suboptimal performance.

In SimFlow, because we manually set $\boldsymbol{\sigma}^2$ as a constant, the entropy term becomes constant as well (see the supplement). The objective now consists of only the reconstruction and generation terms. It then becomes easier to train the entire framework by simply assigning equal weights to both terms.

4.2 Representation alignment

Recent studies [36, 67, 69] show that alignment with features extracted from a representative model significantly benefits the training of a diffusion model. During training, REPA [69] extracts features with a pretrained representation model, such as DINOv2-B [4] and uses a projector to align diffusion hidden states

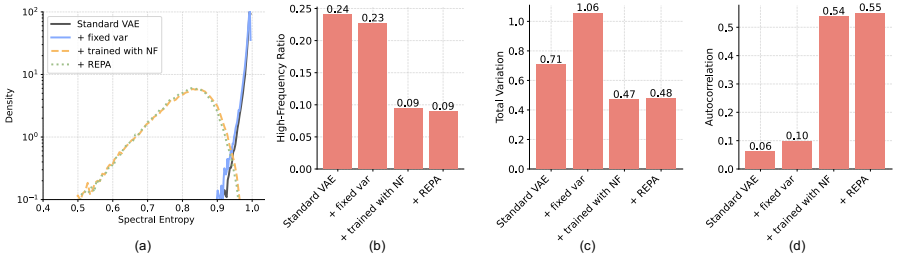

Fig. 4: End-to-end training makes latent space more suitable for developing generative models. (a) Spectral entropy measures the randomness of frequency components; lower values indicate simpler data distributions in the frequency domain. (b) Ratio of high-frequency components. (c) Total variation captures the overall local changes across tokens; lower values imply smoother latents. (d) Autocorrelation reflects how similar a token sequence is to a shifted version of itself; higher autocorrelation indicates stronger spatial consistency.

with DINO features. Besides, VAVAE [67] and REPA-E [36] report stronger benefits of alignment when training VAE or under joint training.

We adopt REPA to SimFlow. As shown in Figure 1(d), we extract features from DINOv2-B and use a projector to align the hidden states of NF with DINO features. Note that, because of the end-to-end nature of SimFlow, REPA applied to SimFlow naturally becomes REPA-E, where REPA gradients flow from the NF to the VAE encoder.

4.3 Why does fixed variance avoid latent collapse?

Leng et al. [36] observe latent space collapse when they naively train DiT and VAE together. We observe the same phenomenon on latent NFs. Intuitively, the key reason is that the loss of generative models encourages latents $\mathbf{x}$ of different images to be close to each other. When latent variance becomes smaller, the MSE loss term for DiTs or NFs is reduced, providing a shortcut for optimization.

As discussed in Section 4.1, our encoder with fixed variance can be considered as a predicted mean $\boldsymbol{\mu}$ plus noise: $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \bar{\sigma}^2\mathbf{I})$. If $\boldsymbol{\mu}$ of different images were close to each other, the differences in $\boldsymbol{\mu}$ would be overwhelmed by noise $\boldsymbol{\epsilon}$, making it hard to reconstruct images. To maintain good reconstruction quality, the system thus tends not to shrink the latent variation as much as naive end-to-end training, allowing for effective end-to-end training. We show similar effects of other noise design in Section 5.3.

Furthermore, we provide a theoretical explanation why naive end-to-end training leads to latent collapse and how SimFlow avoids that (see the supplement).

Proof sketch. In many VAE setups, the KL term is down-weighted (e.g., $\beta = 10^{-6}$), so the objective is dominated by reconstruction and the latent generative loss $\log p_{\boldsymbol{\omega}}(\mathbf{i} \mid \mathbf{x}) + \log p_{\mathrm{NF}}(\mathbf{x}; \boldsymbol{\theta})$. Note that the generative-model loss is typically

dominated by an MSE term, so when the distances between latents become smaller, the loss always decreases. For any given VAE, we can always construct a new VAE whose latents are closer to each other, while keeping the reconstruction loss unchanged and reducing the generative loss (ignoring numerical errors), leading to latent collapse. In contrast, with a fixed variance, there exists an encoder for which making the latents closer does not yield a better loss.

4.4 Working mechanism

Why does a fixed variance improve NF generation? First, our VAE is more robust to imperfect predictions from the generative model. We use different VAEs to reconstruct the 50K validation images in ImageNet 256×256 with increasing noise level on the latent, and then measure the reconstruct quality. Figure 3(a) shows that while a VAE with learnable variance achieves the best reconstruction quality when there is no noise on the latent, VAE with a large and fixed variance is more robust to noise. Thus, when NFs generate imperfect latent, our VAE can still decode good images. **Second**, we linearly interpolate the latents from two images and visualize the reconstruction results of the interpolation points. As seen in Figure 3(b), reconstruction results are better for the VAE with fixed variance, indicating that the latent space of our VAEs is smooth and easy for NFs to learn. Note that, while large variance makes the VAE more robust, it may over-smooth the latent space and limit the reconstruction and generation quality. Thus, a moderate variance performs overall best (see Section 5.3).

How does end-to-end training help? Prior works explore what VAE latent space is suitable for training a generative model, especially for diffusion models [55,66,67]. Based on SimFlow, we present a new perspective: if we jointly train VAE with a generative model, will the latent space become more suitable for generation?

To answer this, we use different statistics to analyze latent space. As shown in Figure 4, the latent space of end-to-end training has lower spectral entropy and less high-frequency components, making it easier for generative models to fit the distribution [55]. Moreover, tokens have lower total variation and higher autocorrelation. This means the token sequence is more suitable for autoregressive modeling. Because our NF is based on AF, the latent space is therefore expected to be more favorable for NF modeling.

4.5 Revisit classifier-free guidance for NFs

CFG [22] significantly improves the performance of diffusion models, by using unconditional outputs to push the class-conditional predictions. This technique has been applied to NFs recently [16,70]. TARFlow [70] does an early exploration but their method lacks theoretical foundation. STARFlow [16] carefully designs CFG for NFs based on mathematical proof, but CFG can only be applied on the last block, limiting the effects on the other blocks.

Table 1: Class-conditional performance on ImageNet 256×256. Methods requiring external pretrained models are highlighted in blue.

Method	Epochs	#Params	VAE		Generation@256 w/o guidance				Generation@256 w/ guidance			
			rFID↓	PSNR↑	gFID↓	IS↑	Prec.↑	Rec.↑	gFID↓	IS↑	Prec.↑	Rec.↑
Pixel Space												
ADM [8]	400	554M	-	-	10.94	101.0	0.69	0.63	3.94	215.8	0.83	0.53
RIN [26]	480	410M	-	-	3.42	182.0	-	-	-	-	-	-
PixelFlow [5]	320	677M	-	-	-	-	-	-	1.98	282.1	0.81	0.60
PixNerd [63]	160	700M	-	-	-	-	-	-	2.15	297.0	0.79	0.59
SiD2 [23]	1280	-	-	-	-	-	-	-	1.38	-	-	-
TARFlow [70]	320	1.4B	-	-	-	-	-	-	4.69	-	-	-
JetFormer [60]	500	2.8B	-	-	-	-	-	-	6.64	-	0.69	0.56
FARMER [74]	320	1.9B	-	-	-	-	-	-	3.60	269.2	0.81	0.51
Autoregressive												
VAR [59]	350	2.0B	-	-	1.92	323.1	0.82	0.59	1.73	350.2	0.82	0.60
MAR [37]	800	943M	0.53	26.18	2.35	227.8	0.79	0.62	1.55	303.7	0.81	0.62
xAR [49]	800	1.1B	0.53	26.18	-	-	-	-	1.24	301.6	0.83	0.64
Latent Diffusion												
DiT [47]	1400	675M	0.61	24.98	9.62	121.5	0.67	0.67	2.27	278.2	0.83	0.57
MaskDiT [75]	1600	675M	0.61	24.98	5.69	177.9	0.74	0.60	2.28	276.6	0.80	0.61
SiT [39]	1400	675M	0.61	24.98	8.61	131.7	0.68	0.67	2.06	270.3	0.82	0.59
MDTv2 [13]	1080	675M	0.61	24.98	-	-	-	-	1.58	314.7	0.79	0.65
REPA [69]	800	675M	0.61	24.98	5.78	158.3	0.70	0.68	1.29	306.3	0.79	0.64
VA-VAE [67]	800	675M	0.28	26.32	2.17	205.6	0.77	0.65	1.35	295.3	0.79	0.65
DDT [64]	400	675M	0.61	24.98	6.27	154.7	0.68	0.69	1.26	310.6	0.79	0.65
REPA-E [36]	800	675M	0.28	26.25	1.69	219.3	0.77	0.67	1.12	302.9	0.79	0.66
RAE [73]	800	839M	0.57	18.86	1.51	242.9	0.79	0.63	1.13	262.6	0.78	0.67
Latent Normalizing Flows												
STARFlow [16]	320	1.4B	2.73	-	-	-	-	-	2.40	-	-	-
SimFlow	160	1.4B	1.21	23.07	13.72	105.2	0.67	0.62	2.15	276.8	0.83	0.57
SimFlow + REPA-E	160	1.4B	1.08	23.17	10.13	124.7	0.71	0.61	1.91	284.4	0.82	0.60

Inspired by the denoising step in TARFlow [70], we run STARFlow CFG first to generate tokens and then derive a score-based step on the generated tokens, *i.e.*,

$$\tilde{\mathbf{x}} = \mathbf{x} + \gamma(\nabla_{\mathbf{x}} \log p_{\mathrm{NF}}(\mathbf{x}|c) - \nabla_{\mathbf{x}} \log p_{\mathrm{NF}}(\mathbf{x}|\phi)), \tag{6}$$

where $\nabla_{\mathbf{x}} \log p_{\mathrm{NF}}(\mathbf{x}|c)$ and $\nabla_{\mathbf{x}} \log p_{\mathrm{NF}}(\mathbf{x}|\phi)$ are the gradients of the NF with and without class labels c, respectively. γ controls the step size. This step leverages the full distribution modeled by the NF and utilizes all NF blocks, rather than only the final one, thereby further improving the generation quality over the CFG method in STARFlow.

5 Experiments

5.1 Implementation details

Experiments are conducted on the ImageNet 256×256 and 512×512 generation tasks [7]. We report gFID [21], IS [53], Precision, and Recall for generation [8]. Besides, rFID, PSNR [25], LPIPS [71], and SSIM [65] are reported for VAE reconstruction evaluation.

We use the MAR VAE architecture [37, 52] and TARFlow as the basic NF model. Both VAE and NF are trained from scratch. Following STARFlow, we use a deep-shallow architecture. Specifically, the NF model has six blocks with 1152 hidden dimensions. The first five blocks have 2 layers while the last deep block has 46 layers. Unless stated otherwise, SimFlow is trained for 80 epochs

Table 2: Class-conditional performance on ImageNet 512×512. Methods requiring external pretrained models are highlighted in blue.

Method	#Params	gFID↓	IS↑	Prec.↑	Rec.↑
BigGAN-deep [3]	158M	8.43	177.9	0.88	0.29
StyleGAN-XL [54]	-	2.41	267.8	0.77	0.52
VAR [59]	2.3B	2.63	303.2	-	-
MAGVIT-v2 [68]	307M	1.91	324.3	-	-
MAR [37]	481M	1.73	279.9	-	-
xAR [49]	608M	1.70	281.5	-	-
ADM [8]	731M	3.85	221.7	0.84	0.53
SiD2	-	1.50	-	-	-
DiT [47]	674M	3.04	240.8	0.84	0.54
SiT [39]	674M	2.62	252.2	0.84	0.57
DiffiT [19]	-	2.67	252.1	0.83	0.55
EDM2 [28]	1.5B	1.25	-	-	-
REPA [69]	675M	2.08	274.6	0.83	0.58
DDT [64]	675M	1.28	305.1	0.80	0.63
RAE [73]	839M	1.13	259.6	0.80	0.63
STARFlow [16]	1.4B	3.00	-	-	-
SimFlow + REPA-E	1.4B	2.74	304.9	0.81	0.57

with a global batch size of 256 and a constant learning rate of 1.0×10^{-4}. In Table 1, we extend the training with extra 80 epochs to get further improved performance, where the learning rate reduces from 1.0×10^{-4} to 1.0×10^{-6} at a cosine rate like STARFlow. Details can be found in the supplement.

5.2 Main evaluation

Comparison with state-of-the-art methods on ImageNet 256×256. Results are shown in Table 1. SimFlow achieves a gFID of 2.15, significantly outperforming the prior NF method STARFlow (gFID=2.40). Note that STARFlow is trained for 320 epochs while SimFlow is trained for 160 epochs. Our jointly-trained VAE has better reconstruction quality (rFID=1.21) than STARFlow (rFID=2.73). Besides, SimFlow surpasses a representative diffusion model, DiT [47]. Although SimFlow needs more parameters, the convergence rate is more than 8 times faster than DiT.

With REPA-E, SimFlow establishes the new state-of-the-art performance among NFs (gFID 1.91). REPA-E not only improves the generation quality in terms of gFID but also the reconstruction performance of VAE, with better rFID and PSNR. It indicates that the guidance from a pretrained model is helpful for building a better latent space, benefiting both the training of VAEs and NFs.

Comparison with state-of-the-art methods on ImageNet 512×512. Results are summarized in Table 2. SimFlow with REPA-E also works on higher resolution images and significantly surpasses STARFlow. Sample images generated from SimFlow are shown in Figure 5.

Effectiveness of end-to-end training. We present results in Figure 6. Specifically, we train a normal VAE with fixed variance and then train the NF

Fig. 5: Qualitative Results. We show selected examples of class-conditional generation on ImageNet using SimFlow.

model without updating the VAE at all, denoted as 'Fixed var' under the 'Frozen VAE' category. It is clear that our method 'Fixed var' under 'End-to-end' training significantly outperforms the other (gFID = 16.97 vs. gFID 67.41). Only setting a fixed variance does not improve the latent space, but the end-to-end training makes the latent space more suitable for generation. Moreover, we apply noise-augmented training [16] with encoder frozen (as in STARFlow) and end-to-end training, respectively. As seen in Figure 6, end-to-end training also obtains lower gFID, indicating its effectiveness.

Comparing fixed variance with noise augmentation under end-to-end training. As shown in Figure 6, fixed variance yields lower gFID (16.97 vs. 20.01). This demonstrates that fixing variance is a useful technique: it simplifies the pipeline while having competitive performance. Here, we do not claim our method is superior to noise augmentation because they essentially share similar principles, *i.e.*, perturbing the latents. We speculate that noise augmentation would have close performance in end-to-end training if we train the model longer or tune the hyperparameters.

Reconstruction vs. generation. The best reconstruction results are achieved by learnable variance (see the supplement). However, Figure 6 shows that learnable variance does not necessarily mean better generation. Our observation is consistent with prior works [67, 73], but from a different perspective of fixing or training the variance term in VAE.

Comparing the revised CFG method with STARFlow CFG [16]. We run STARFlow CFG first to generate tokens and then derive a score-based step, Eq. (6) on the tokens. As demonstrated in Figure 7, with $\gamma = 0.25$, our CFG significantly improves the optimal generation quality. Besides, it introduces only a mild increase in inference time compared to STARFlow (see the supplement).

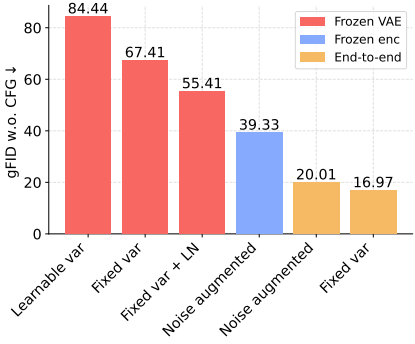

Fig. 6: Variant studies. Fixing the variance and end-to-end training can significantly improve the generation quality of the latent NF model.

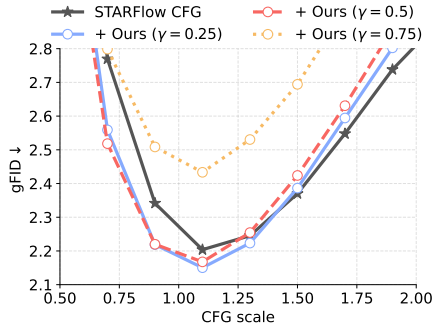

Fig. 7: CFG method comparison. Our CFG method in Eq. 6, with an appropriate hyperparameter γ, improves the best point over the original STARFlow CFG [16].

Table 3: Comparison of different latent dimensions. A low latent dimension limits reconstruction while a high dimension makes it harder for NFs to learn the latent space. The best results are shown in bold, and the default setting is highlighted in red.

Dim	rFID↓	PSNR↑	LPIPS↓	SSIM↑	gFID w.o. CFG↓	gFID w. CFG↓
16	4.98	19.94	0.327	0.45	**11.00**	3.55
32	2.60	21.42	0.274	0.51	12.77	2.61
64	1.49	23.01	0.222	0.59	21.44	**2.53**
128	**0.86**	**24.61**	**0.183**	**0.66**	33.43	3.62

5.3 Further analysis

Comparing different latent dimensions. Table 3 presents the performance of different VAE latent dimensions. Because we enforce a large and constant variance ($\bar{\sigma}^2 = 0.5^2$), a low latent dimension such as 16 cannot preserve sufficient information. In contrast, a high dimension such as 128, while benefiting reconstruction, would make it challenging for NF modeling. The best trade-off is 64-dimensional.

Comparing different variance values. From Table 4, $\bar{\sigma}^2 = 0.5^2$ generally results in good reconstruction and generation quality. We notice that larger variance accelerates convergence because the latent space is smoother, but reduces the upper bound of the final performance. Overall, SimFlow performs well across a wide range of variance.

Comparing different model sizes. In Table 5, scaling up NFs improves both the reconstruction and generation quality. A small NF cannot fit a complex latent distribution, and thus, the encoder tends to maintain a simple latent space, sacrificing the reconstruction quality. When the NF is larger and more

Table 4: Comparing different levels of variance. A moderate variance $\bar{\sigma}^2 = 0.5^2$ performs the best. The superior results are shown in bold, and the default setting is highlighted in yellow.

Var	rFID↓	PSNR↑	LPIPS↓	SSIM↑	gFID w.o. CFG↓	gFID w. CFG↓
0.1^2	1.90	22.61	0.238	0.57	25.54	3.01
0.25^2	1.55	22.94	0.225	0.58	21.35	2.57
0.5^2	**1.49**	**23.01**	**0.222**	**0.59**	21.44	2.53
0.75^2	1.54	22.88	0.224	**0.59**	**17.76**	**2.39**
1.0	1.76	22.60	0.231	0.58	19.36	2.66

Table 5: Comparing different model sizes. Large NFs improve both the reconstruction and generation quality. The best results are shown in bold, and the default setting is highlighted in gray.

Model	#Params	rFID↓	PSNR↑	LPIPS↓	SSIM↑	gFID w.o. CFG↓	gFID w. CFG↓
S	37M	1.70	22.81	0.230	0.58	65.57	15.34
B	141M	1.66	22.89	0.226	0.58	52.77	9.72
L	475M	1.52	22.95	0.224	**0.59**	33.53	3.72
XL	695M	1.51	22.99	0.224	**0.59**	28.60	3.51
XXL	1.4B	**1.49**	**23.01**	**0.222**	**0.59**	**21.44**	**2.53**

Table 6: Comparison of different ways of adding noise to latent x. All these ways enable end-to-end training. Our method 'Additive' performs the best and is highlighted in green.

Noise	Equation	rFID↓	PSNR↑	gFID w.o. CFG↓
No noise	$\mathbf{x}' = \mathbf{x}$	Collapse		
Linear	$\mathbf{x}' = t\mathbf{x} + (1-t)\boldsymbol{\epsilon}$	8.54	19.74	71.0
Slerp	$\mathbf{x}' = t\mathbf{x} + \sqrt{1-t^2}\boldsymbol{\epsilon}$	5.56	20.63	49.7
Additive	$\mathbf{x}' = \mathbf{x} + \boldsymbol{\epsilon}$	2.90	22.22	39.6

Table 7: Text-to-image generation. Fixing variance significantly improves generation quality across different latent dimensions.

Latent dim	Var	FID-25K↓	IS↑
16/32/64	Learnable	Latent collapse	
STARFlow reproduce		68.85	10.82
16	0.1^2	56.08 (-12.77)	11.18 (+0.36)
16	0.5^2	36.46 (-32.39)	13.97 (+3.15)
16	1.0^2	34.48 (-34.37)	14.19 (+3.37)
32	0.1^2	60.98 (-7.87)	12.10 (+1.28)
32	0.5^2	47.38 (-21.47)	13.16 (+2.34)
32	1.0^2	41.63 (-27.22)	12.24 (+1.42)
64	0.1^2	79.16 (+10.31)	11.40 (+0.58)
64	0.5^2	57.01 (-11.84)	11.99 (+1.17)
64	1.0^2	53.52 (-15.33)	12.24 (+1.42)

Table 8: Applying fixed variance to end-to-end training of other generative models and VAEs. The colored rows apply fixed variance.

VAE		Gen.		Joint	FID		
Dim	Train	Model	Train		w/o CFG ↓	Best CFG ↓	Best setting
16	Unk.	MAR-B	80ep	✗	16.85	3.46	cfg=4.0
16	Unk.+80ep	MAR-B	80ep	✓	19.15	3.33	cfg=3.0
32	50ep	SiT-XL/1	50ep	✗	20.36	4.41	cfg=3.1, gh=0.75
32	50ep	SiT-XL/1	50ep	✓	17.70	4.24	cfg=3.35, gh=0.35
64	50ep	SiT-XL/1	50ep	✗	28.33	5.67	cfg=3.6, gh=0.825
64	50ep	SiT-XL/1	50ep	✓	19.11	4.21	cfg=2.5, gh=0.55
32	80ep	REPA-E	80ep	✗	6.30	1.66	cfg=2.05, gh=0.8
32	80ep	REPA-E	80ep	✓	4.38	1.56	cfg=2.1, gh=0.525

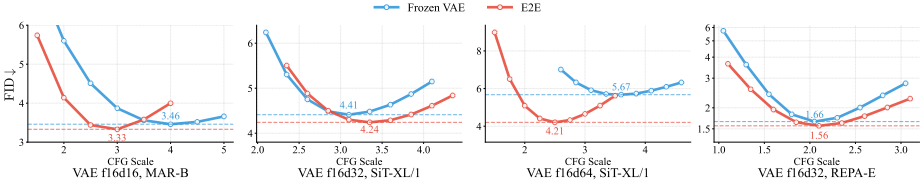

Fig. 8: CFG sweep for other generative models. Across MAR, SiT, and REPA-E backbones, end-to-end training consistently improves the best FID over the two-stage baseline under CFG sweeps.

expressive, generation quality is improved. Meanwhile, the encoder can embed more information into latents, enhancing reconstruction quality as well.

Comparison of different ways of adding noise to latent x. We test different latent noise discussed in prior works [42, 48, 66]. Specifically, we train VAEs with NFs for 100K iterations with these perturbations. As we expect in Section 4.4, Table 6 shows that all these methods avoid collapse during end-to-end training. Note that these perturbations have different strengths in removing information in latents, leading to different performance. If we tune the weights of the strengths of these methods, they should lead to similar results, which we leave for future work.

Effect of fixed variance on text-to-image (T2I) generation. We also conduct preliminary experiments on the T2I generation task. Specifically, we train models on COCO train2017 at 256×256 with STARFlow-base and the FLAN-T5-XL text encoder. The STARFlow baseline uses the pretrained SD-VAE ($f8d4$) with additive latent noising ($var = 0.3^2$). All runs use batch size 32, learning rate 0.0001, 100K steps, and the same loss config as SimFlow. Evaluation uses val2017 captions, optimal CFG scale (CFG scale actually has minimal effect on results), EMA weights, and reports FID-25K against the val2017 set. As shown in Table 7, fixing the variance stabilizes training and consistently improves generation quality.

Effectiveness of SimFlow applied to other generative models. We perform a preliminary study by jointly training VAE with MAR [37] or SiT [39]. For MAR, we choose the VAE provided by Li et al. [37] and train the MAR-B model for 80 epochs, with a batch size of 1024. For SiT, we choose the VAE in LDM [52] and use the 32- and 64-dimensional checkpoints from Yao et al. [67], which were trained for 50 epochs. We compare 1) SiT trained for 50 epochs with the pretrained VAE frozen, and 2) jointly train SiT and a new VAE for 50 epochs from scratch. All models are evaluated with EMA, and we tune CFG with a coarse-to-fine search. We first perform a rough search using large step sizes (e.g., 0.5). After locating a promising range, we progressively narrow the search interval and reduce the step size. For SiT, we search CFG with another sampling hyperparameter "guidance-high (gh)".

Table 8 and Figure 8 shows our method also stabilizes the training of other generative models such as mask autoregressive model or diffusion models, and we did not observe latent collapse during training. Besides, end-to-end training achieves better generation quality after the same iterations. We notice that a recent work also adopts a fixed variance to stabilize the end-to-end training of the autoencoder and diffusion model [20]. Thus, end-to-end training for different generative models with fixed-variance VAEs is a promising direction.

Generalization to other high-resolution datasets and modalities. We conduct additional experiments on two high-resolution datasets: CelebA-HQ 1024×1024 [27] and AFHQv2 512×512 [6], as well as a text-to-audio generation task [30,38]. For each dataset, we compare the standard two-stage pipeline, which first trains a VAE and then trains the NF on frozen VAE latents, against our end-to-end SimFlow training. The results in the supplement further demonstrate the effectiveness of SimFlow.

6 Conclusion

This paper presents SimFlow, an end-to-end training framework for latent NFs by simply fixing the VAE variance. This makes latent space smoother and helps NFs generalize better when sampling, without needing extra noise schedules or denoising steps. Experiments show that SimFlow improves generation quality and speeds up training compared to existing NF methods. Future work will expand this framework to large-scale text-to-image training and explore a second-stage training with the VAE fixed after joint training.

Acknowledgments

We sincerely thank Caixia Zhou, Hanhong Zhao, and Shuyang Gu for their valuable support. We also thank Ming Xu, Jaskirat Singh, Yunzhong Hou, Weijian Deng, and our colleagues at the ANU lab for their helpful feedback. We are also grateful to our colleagues at ByteDance Seed for the insightful discussions during this project. The ANU component of this work was supported by an Australian Research Council (ARC) Linkage Grant (project number LP210200931).

References

1. Bao, F., Nie, S., Xue, K., Cao, Y., Li, C., Su, H., Zhu, J.: All are worth words: A vit backbone for diffusion models. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 22669–22679 (2023)
2. Bhattacharyya, A., Hanselmann, M., Fritz, M., Schiele, B., Straehle, C.N.: Conditional flow variational autoencoders for structured sequence prediction. arXiv preprint arXiv:1908.09008 (2019)
3. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: Int. Conf. Learn. Represent. (2019)
4. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Int. Conf. Comput. Vis. pp. 9650–9660 (2021)
5. Chen, S., Ge, C., Zhang, S., Sun, P., Luo, P.: Pixelflow: Pixel-space generative models with flow. arXiv preprint arXiv:2504.07963 (2025)
6. Choi, Y., Uh, Y., Yoo, J., Ha, J.W.: Stargan v2: Diverse image synthesis for multiple domains. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 8188–8197 (2020)
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 248–255. Ieee (2009)
8. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. Adv. Neural Inform. Process. Syst. **34**, 8780–8794 (2021)
9. Dinh, L., Krueger, D., Bengio, Y.: Nice: Non-linear independent components estimation. arXiv preprint arXiv:1410.8516 (2014)
10. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real nvp. In: Int. Conf. Learn. Represent. (2017)
11. Draxler, F., Sorrenson, P., Zimmermann, L., Rousselot, A., Köthe, U.: Free-form flows: Make any architecture a normalizing flow. In: International Conference on Artificial Intelligence and Statistics. pp. 2197–2205. PMLR (2024)
12. Draxler, F., Wahl, S., Schnoerr, C., Koethe, U.: On the universality of volume-preserving and coupling-based normalizing flows. In: Int. Conf. Machine Learn. pp. 11613–11641. PMLR (2024)
13. Gao, S., Zhou, P., Cheng, M.M., Yan, S.: Mdtv2: Masked diffusion transformer is a strong image synthesizer. arXiv preprint arXiv:2303.14389 (2023)
14. Giaquinto, R., Banerjee, A.: Gradient boosted normalizing flows. Adv. Neural Inform. Process. Syst. **33**, 22104–22117 (2020)
15. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Adv. Neural Inform. Process. Syst. **27** (2014)
16. Gu, J., Chen, T., Berthelot, D., Zheng, H., Wang, Y., Zhang, R., Dinh, L., Bautista, M.A., Susskind, J., Zhai, S.: Starflow: Scaling latent normalizing flows for high-resolution image synthesis. Adv. Neural Inform. Process. Syst. **38**, 120986–121022 (2026)
17. Gu, J., Shen, Y., Chen, T., Dinh, L., Wang, Y., Bautista, M.A., Berthelot, D., Susskind, J., Zhai, S.: Starflow-v: End-to-end video generative modeling with autoregressive normalizing flows. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 9084–9094 (2026)
18. Han, C., Li, G., Wu, J., Sun, Q., Cai, Y., Peng, Y., Ge, Z., Zhou, D., Tang, H., Zhou, H., Liu, K., Xia, S.T., Jiao, B., Jiang, D., Zhang, X., Zhu, Y.: Nextstep-1: Toward autoregressive image generation with continuous tokens at scale. In: Int. Conf. Learn. Represent. (2026)

19. Hatamizadeh, A., Song, J., Liu, G., Kautz, J., Vahdat, A.: Diffit: Diffusion vision transformers for image generation. In: Eur. Conf. Comput. Vis. pp. 37–55. Springer (2024)
20. Heek, J., Hoogeboom, E., Mensink, T., Salimans, T.: Unified latents (ul): How to train your latents. arXiv preprint arXiv:2602.17270 (2026)
21. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Adv. Neural Inform. Process. Syst. **30** (2017)
22. Ho, J., Salimans, T.: Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598 (2022)
23. Hoogeboom, E., Mensink, T., Heek, J., Lamerigts, K., Gao, R., Salimans, T.: Simpler diffusion: 1.5 fid on imagenet512 with pixel-space diffusion. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 18062–18071 (June 2025)
24. Huang, C.W., Krueger, D., Lacoste, A., Courville, A.: Neural autoregressive flows. In: Int. Conf. Machine Learn. pp. 2078–2087. PMLR (2018)
25. Huynh-Thu, Q., Ghanbari, M.: Scope of validity of psnr in image/video quality assessment. Electronics letters **44**(13), 800–801 (2008)
26. Jabri, A., Fleet, D.J., Chen, T.: Scalable adaptive computation for iterative generation. In: Int. Conf. Machine Learn. pp. 14569–14589. PMLR (2023)
27. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. In: Int. Conf. Learn. Represent. (2018)
28. Karras, T., Aittala, M., Lehtinen, J., Hellsten, J., Aila, T., Laine, S.: Analyzing and improving the training dynamics of diffusion models. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 24174–24184 (2024)
29. Ke, G., XUE, H.: Hyperspherical latents improve continuous-token autoregressive generation. In: Int. Conf. Learn. Represent. (2026)
30. Kim, C.D., Kim, B., Lee, H., Kim, G.: Audiocaps: Generating captions for audios in the wild. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 119–132 (2019)
31. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
32. Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. Adv. Neural Inform. Process. Syst. **31** (2018)
33. Kingma, D.P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., Welling, M.: Improved variational inference with inverse autoregressive flow. Adv. Neural Inform. Process. Syst. **29** (2016)
34. Kobyzev, I., Prince, S.J., Brubaker, M.A.: Normalizing flows: An introduction and review of current methods. IEEE Trans. Pattern Anal. Mach. Intell. **43**(11), 3964–3979 (2020)
35. Kouzelis, T., Kakogeorgiou, I., Gidaris, S., Komodakis, N.: Eq-vae: Equivariance regularized latent space for improved generative image modeling. In: Int. Conf. Machine Learn. pp. 31648–31666. PMLR (2025)
36. Leng, X., Singh, J., Hou, Y., Xing, Z., Xie, S., Zheng, L.: Repa-e: Unlocking vae for end-to-end tuning of latent diffusion transformers. In: Int. Conf. Comput. Vis. pp. 18262–18272 (2025)
37. Li, T., Tian, Y., Li, H., Deng, M., He, K.: Autoregressive image generation without vector quantization. Adv. Neural Inform. Process. Syst. **37**, 56424–56445 (2024)
38. Liu, H., Huang, R., Liu, Y., Cao, H., Wang, J., Cheng, X., Zheng, S., Zhao, Z.: Audiolcm: Efficient and high-quality text-to-audio generation with minimal inference

steps. In: Proceedings of the 32nd ACM International Conference on Multimedia. p. 7008–7017. Association for Computing Machinery (2024)
39. Ma, N., Goldstein, M., Albergo, M.S., Boffi, N.M., Vanden-Eijnden, E., Xie, S.: Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In: Eur. Conf. Comput. Vis. pp. 23–40. Springer (2024)
40. Máté, B., Klein, S., Golling, T., Fleuret, F.: Flowification: Everything is a normalizing flow. Adv. Neural Inform. Process. Syst. **35**, 35478–35489 (2022)
41. Mathieu, E., Nickel, M.: Riemannian continuous normalizing flows. Adv. Neural Inform. Process. Syst. **33**, 2503–2515 (2020)
42. Meshchaninov, V., Chimbulatov, E., Shabalin, A., Abramov, A., Vetrov, D.: Cosmos: Compressed and smooth latent space for text diffusion modeling. In: Adv. Neural Inform. Process. Syst. (2026)
43. Pang, Z., Zhang, T., Luan, F., Man, Y., Tan, H., Zhang, K., Freeman, W.T., Wang, Y.X.: Randar: Decoder-only autoregressive visual generation in random orders. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 45–55 (2025)
44. Papamakarios, G., Nalisnick, E., Rezende, D.J., Mohamed, S., Lakshminarayanan, B.: Normalizing flows for probabilistic modeling and inference. Journal of Machine Learning Research **22**(57), 1–64 (2021)
45. Papamakarios, G., Pavlakou, T., Murray, I.: Masked autoregressive flow for density estimation. Adv. Neural Inform. Process. Syst. **30** (2017)
46. Patacchiola, M., Shysheya, A., Hofmann, K., Turner, R.E.: Transformer neural autoregressive flows. arXiv preprint arXiv:2401.01855 (2024)
47. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Int. Conf. Comput. Vis. pp. 4195–4205 (2023)
48. Qiu, K., Li, X., Chen, H., Kuen, J., Xu, X., Gu, J., Luo, Y., Raj, B., Lin, Z., Savvides, M.: Image tokenizer needs post-training. arXiv preprint arXiv:2509.12474 (2025)
49. Ren, S., Yu, Q., He, J., Shen, X., Yuille, A., Chen, L.C.: Beyond next-token: Next-x prediction for autoregressive visual generation. In: Int. Conf. Comput. Vis. pp. 15781–15791 (2025)
50. Ren, S., Yu, Q., He, J., Shen, X., Yuille, A., Chen, L.C.: Flowar: Scale-wise autoregressive image generation meets flow matching. In: Int. Conf. Machine Learn. pp. 51489–51502. PMLR (2025)
51. Rezende, D., Mohamed, S.: Variational inference with normalizing flows. In: Int. Conf. Machine Learn. pp. 1530–1538. PMLR (2015)
52. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 10684–10695 (2022)
53. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. Adv. Neural Inform. Process. Syst. **29** (2016)
54. Sauer, A., Schwarz, K., Geiger, A.: Stylegan-xl: Scaling stylegan to large diverse datasets. In: ACM SIGGRAPH 2022 conference proceedings. pp. 1–10 (2022)
55. Skorokhodov, I., Girish, S., Hu, B., Menapace, W., Li, Y., Abdal, R., Tulyakov, S., Siarohin, A.: Improving the diffusability of autoencoders. In: Int. Conf. Machine Learn. pp. 55876–55905. PMLR (2025)
56. Su, J., Wu, G.: f-vaes: Improve vaes with conditional flows. arXiv preprint arXiv:1809.05861 (2018)
57. Sun, P., Jiang, Y., Chen, S., Zhang, S., Peng, B., Luo, P., Yuan, Z.: Autoregressive model beats diffusion: Llama for scalable image generation. arXiv preprint arXiv:2406.06525 (2024)

58. Sun, Y., Bao, H., Wang, W., Peng, Z., Dong, L., Huang, S., Chang, Y., Wang, J., Wei, F.: Multimodal latent language modeling with next-token diffusion. In: Int. Conf. Machine Learn. (2026)
59. Tian, K., Jiang, Y., Yuan, Z., Peng, B., Wang, L.: Visual autoregressive modeling: Scalable image generation via next-scale prediction. Adv. Neural Inform. Process. Syst. **37**, 84839–84865 (2024)
60. Tschannen, M., Pinto, A.S., Kolesnikov, A.: Jetformer: An autoregressive generative model of raw images and text. In: Int. Conf. Learn. Represent. (2025)
61. Vahdat, A., Kreis, K., Kautz, J.: Score-based generative modeling in latent space. Adv. Neural Inform. Process. Syst. **34**, 11287–11302 (2021)
62. Van Den Oord, A., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. In: Int. Conf. Machine Learn. PMLR (2016)
63. Wang, S., Gao, Z., Zhu, C., Huang, W., Wang, L.: Pixnerd: Pixel neural field diffusion. In: Int. Conf. Learn. Represent. (2026)
64. Wang, S., Tian, Z., Huang, W., Wang, L.: Ddt: Decoupled diffusion transformer. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 40633–40642 (2026)
65. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
66. Yang, J., Li, T., Fan, L., Tian, Y., Wang, Y.: Latent denoising makes good tokenizers. In: Int. Conf. Learn. Represent. (2026)
67. Yao, J., Yang, B., Wang, X.: Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 15703–15712 (2025)
68. Yu, L., Lezama, J., Gundavarapu, N.B., Versari, L., Sohn, K., Minnen, D., Cheng, Y., Gupta, A., Gu, X., Hauptmann, A.G., Gong, B., Yang, M.H., Essa, I., Ross, D.A., Jiang, L.: Language model beats diffusion - tokenizer is key to visual generation. In: Int. Conf. Learn. Represent. (2024)
69. Yu, S., Kwak, S., Jang, H., Jeong, J., Huang, J., Shin, J., Xie, S.: Representation alignment for generation: Training diffusion transformers is easier than you think. In: Int. Conf. Learn. Represent. (2025)
70. Zhai, S., Zhang, R., Nakkiran, P., Berthelot, D., Gu, J., Zheng, H., Chen, T., Bautista, M.Á., Jaitly, N., Susskind, J.M.: Normalizing flows are capable generative models. In: Int. Conf. Machine Learn. pp. 74348–74369. PMLR (2025)
71. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 586–595 (2018)
72. Zhang, R., Zhai, S., Gu, J., Zhang, Y., Zheng, H., Chen, T., Bautista, M.A., Susskind, J., Jaitly, N.: Flexible language modeling in continuous space with transformer-based autoregressive flows. Adv. Neural Inform. Process. Syst. **38**, 173273–173323 (2026)
73. Zheng, B., Ma, N., Tong, S., Xie, S.: Diffusion transformers with representation autoencoders. In: Int. Conf. Learn. Represent. (2026)
74. Zheng, G., Zhao, Q., Yang, T., Xiao, F., Lin, Z., Wu, J., Deng, J., Zhang, Y., Zhu, R.: Farmer: Flow autoregressive transformer over pixels. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 25730–25741 (2026)
75. Zheng, H., Nie, W., Vahdat, A., Anandkumar, A.: Fast training of diffusion models with masked transformers. Transactions on Machine Learning Research (2024)