Guiding a Diffusion Transformer with the Internal Dynamics of Itself

Supplementary Material

In this file, we provide more implementation and experimental details which are not included in the main text. In Section A, we provide the details of all the evaluation metrics and further compare the generation performance under uniform sampling. In Section B, we provide a detailed comparison of computational resource consumption during the sampling stage between the proposed IG and REPA. In Section C, we provide a comparison of the generate results under different sampling guidance methods. In Section D, we provide the experimental results of the proposed IG on the 512 $\times$ 512 ImageNet dataset. In Section E, we provide more hyperparameter settings and implementation details for Diffusion Transformer experiments of various scales. In Section F, We provide hyperparameter settings and implementation details for the 2D toy example. In Section G, We provide a more detailed discussion on the related work of our approach. In Section H, we provide more uncurated generation results on ImageNet 256×256 from the SiT-XL+IG with CFG.

Table 1. **Class-conditional performance on ImageNet 256×256 with balanced sampling.** LightningDiT + IG reaches an FID of 1.24 without CFG, outperforming all prior state-of-the-art VQ-based and Diffusion-based methods by a large margin, and achieves an FID of 1.07 with CFG. For a fair comparison, all prior methods employ **uniform balanced sampling** 50K samples across 1,000 class labels.

Method	Epochs	#Params	w/o CFG				w/ CFG or Autoguidance			
			FID↓	IS↑	Prec.↑	Rec.↑	FID↓	IS↑	Prec.↑	Rec.↑
Autoregressive										
VAR [51]	350	2.0B	1.92	**323.1**	**0.82**	0.59	1.73	**350.2**	0.82	0.60
MAR [37]	800	943M	2.35	227.8	0.79	0.62	1.55	303.7	0.81	0.62
xAR [44]	800	1.1B	-	-	-	-	1.24	301.6	**0.83**	0.64
Latent Diffusion with Self-supervised Representation Model										
REPA [60]	800	675M	5.78	158.3	0.70	0.68	1.29	306.3	0.79	0.64
DDT [55]	400	675M	6.27	154.7	0.68	**0.69**	1.26	310.6	0.79	0.65
REPA-E [34]	800	675M	1.70	217.3	0.77	0.66	1.15	304.0	0.79	0.66
LightningDiT [59]	800	675M	2.11	207.0	0.77	0.65	1.28	300.5	0.80	0.65
RAE (DiT^{DH}) [62]	800	839M	1.51	242.9	0.79	0.63	1.13	262.6	0.78	**0.67**
LightningDiT+IG (ours)	680	678M	**1.24**	229.6	0.78	0.66	**1.07**	274.1	0.79	0.66

A. Evaluation Metric

We strictly follow the setup and use the same reference batches of ADM [6] for evaluation, following their official implementation. We use NVIDIA A6000 pro GPUs or 4090Ti GPUs for evaluation and enable tf32 precision for faster generation. In what follows, we explain the main concept of metrics that we used for the evaluation.

- **FID** [16] measures the feature distance between the distributions of real and generated images. It uses the Inception-v3 network [50] and computes distance based on an assumption that both feature distributions are multivariate gaussian distributions.
- **sFID** [40] proposes to compute FID with intermediate spatial features of the Inception-v3 network to capture the generated images' spatial distribution.
- **IS** [47] also uses the Inception-v3 network but use logit for evaluation of the metric. Specifically, it measures a KL-divergence between the original label distribution and the distribution of logits after the softmax normalization.
- **Precision and recall** [31] are based on their classic definitions: the fraction of realistic images and the fraction of training data manifold covered by generated data.

As stated in [62], using class-balanced sampling can lead to better results compared to random sampling because it more closely approximates the true label distribution of the training set. As the FID value approaches a lower range, these subtle differences begin to have a greater impact. The vast majority of VQ-based generation methods [37, 44, 51] have adopt the balanced sampling. To conduct a comprehensive and fair comparison with more baselines, we also adopt balanced sampling to conduct a comprehensive comparison of the current state-of-the-art VQ and Diffusion-based methods, as shown in Table 1.

B. Computational Cost Comparison

We compare the computational efficiency of SiT-XL/2+REPA [60] and SiT-XL+IG under the same model scale in Table 2. IG introduces only a marginal increase in parameter count and FLOPs relative to SiT-XL, while maintaining neral identical latency. Despite the minimal computational overhead, IG yields substantial improvements in generation quality, achieving a relative reduction in FID, alongside an increase in IS. These results demonstrate that IG simultaneously improves generation quality and computational efficiency, highlighting its effectiveness as a general-purpose enhancement for generative models.

Table 2. **Computational cost and performance comparison.** This table compares REPA and IG on ImageNet 256×256, detailing model size, FLOPs, sampling steps, latency, and generation quality metrics. the proposed IG achieves substantially better sample quality with negligible increases in computational cost.

Method	#Params	FLOPs↓	Latency (s)↓	FID↓	IS↑
SiT-XL/2 + REPA	675	114.46	6.18	5.90	157.8
SiT-XL/2 + IG	678 (+0.44%)	114.47 (+0.01%)	6.19 (+0.16%)	1.75 (-70.34%)	228.6 (+44.87%)

C. Comparison of Guidance Methods

To further illustrate the advantages of our proposed IG, we compared it with the original CFG [17] and Autoguidance [26] methods. For CFG method, we use the vanilla LightningDiT-XL/1 model which is trained for 800 epochs and employed its official CFG coefficients. For the Autoguidance method, since it is difficult to determine the optimal bad version model, we initialized a 12-layer LightningDiT model and trained it for 10 epochs. We roughly searched for the optimal coefficients of Autoguidance and determined it to be 1.15. We also compare our IG with original Autoguidance on the vanilla EDM2-S [28] model on ImageNet 64 dataset with the same training setting, which is shown in Table 3. As shown in Table 4, our IG method outperforms both adding only CFG or only adding Autoguidance. Moreover, further combining CFG can achieve the current state-of-the-art in image generation, and has a significant improvement compared to other guidance methods.

Table 3. Comparision with original Autoguidance on the EDM2-S model on ImageNet 64 dataset.

Model	EDM2-S	EDM2-S+AutoGuidance	EDM2-S+IG
FID↓	1.58	1.01	**0.99**

Table 4. **Different guidance strategies comparison.** This table compares various sampling guidance methods. We compare the results of the standard CFG, Autoguidance, as well as our proposed IG and the combination of IG and CFG. Our proposed IG is more flexible compared to Autoguidance and can significantly enhance the results when combined with CFG as a plug-and-play approach.

Method	Epochs	**FID↓**	**sFID↓**	**IS↑**	**Prec.↑**	**Rec.↑**
LightningDiT-XL/1 + CFG [59]	800	1.35	4.15	**295.3**	0.79	0.65
LightningDiT-XL/1 + Autoguidance [26]	680	1.81	4.06	215.9	0.79	0.63
LightningDiT-XL/1 + IG (ours)	680	1.34	**3.94**	229.3	0.78	0.65
LightningDiT-XL/1 + IG + CFG (ours)	680	**1.19**	4.11	269.0	**0.79**	**0.66**

D. 512×512 ImageNet

To further validate IG's effectiveness, we conduct experiments at 512×512 resolution following REPA's protocol [60] with Muon optimizer [22]. The RGB images are processed through the VAE [45] to yield 64×64×3 latents. As demonstrated in Table, IG surpasses the performance of REPA trained for 200 epochs and SiT-XL/2 trained for 600 epochs in terms of FID at only 60 epochs, demonstrating its superior effectiveness.

E. Hyperparameter and More Implementation Details

More implementation details. We implement our models based on the original DiT, SiT and LightningDiT implementation. To speed up training and save GPU memory, we use mixed-precision (fp16) with gradient clipping and FlashAttention [5] operation for attention computation. We also pre-compute compressed latent vectors from raw pixels via stable diffusion VAE and VA-VAE and use these latent vectors. We do not apply any data augmentation as discussed in MaskDiT and REPA. For stable diffusion VAE, we use `stabilityai/sd-vae-ft-ema` for encoding images to latent vectors and decoding latent vectors to images. For the output head, we use a Linear layer to map the feature to the output size.

Figure 1. **Samples on ImageNet 512×512** from SiT-XL/2 + IG (1.4) using CFG with $w = 1.8$.

Table 5. **Performance comparison** on ImageNet 512×512.

Model	Epochs	FID↓	sFID↓	IS↑	Pre.↑	Rec.↑
Pixel diffusion						
VDM++	-	2.65	-	278.1	-	-
ADM-G, ADM-U	400	2.85	5.86	221.7	0.84	0.53
Simple diffusion (U-Net)	800	4.28	-	171.0	-	-
Simple diffusion (U-ViT, L)	800	4.53	-	205.3	-	-
Latent diffusion, Transformer						
MaskDiT	800	2.50	5.10	256.3	0.83	0.56
DiT-XL/2	600	3.04	5.02	240.8	0.84	0.54
SiT-XL/2	600	2.62	4.18	252.2	0.84	0.57
+ REPA	80	2.44	4.21	247.3	0.84	0.56
+ REPA	100	2.32	4.16	255.7	0.84	0.56
+ REPA	200	2.08	4.19	274.6	0.83	0.58
+ IG	**60**	**1.78**	**3.92**	**286.3**	0.81	**0.62**

The use of Muon In our large-scale experiment with the LightningDiT-XL/1 model, we adopt the Muon optimizer [22]. We find that in the environment of PyTorch, when using the pre-trained self-supervised representation model (such as DINOv2-B [41]) to align the encoder of the VAE, there will be unstable training issues in the early stage of training. After careful investigation and experimentation, we discover that replacing the AdamW optimizer with Muon could effectively alleviate such issue. Compared to AdamW, the Muon optimizer has a certain acceleration convergence effect in the early stage of model training, but we observe that the gap between them gradually narrowed as the training time increased, reaching similar final convergence results. In the later stage of training, there is a small probability of encountering training instability issues again.

Resuming the training with full precision can solve this problem.

Computing resources. We use NVIDIA A6000 pro 96 GB GPUs for training largest model (LightningDiT-XL/1, SiT-XL/2); and uses NVIDIA 4090 24GB GPUs for training smaller model (L, B). When sampling, we use either NVIDIA A6000 pro 96 GB GPUs or NVIDIA RTX 4090 24GB GPUs to obtain the samples for evaluation.

Table 6. **Defualt hyperparameter setup.** Unless other otherwise specified, we use these sets of hyperparameters for different models. In our ablation experiments, settings are also kept the same except those we point out in Table 6.

	SiT-B	SiT-L	SiT-XL	DiT-B	DiT-L	LightningDiT-XL
Architecture						
Input dim.	32×32×4	32×32×4	32×32×4	32×32×4	32×32×4	16×16×32
Patch size	2	2	2	2	2	2
Num. layers	12	24	28	12	24	28
Hidden dim.	768	1024	1152	768	1024	1152
Num. heads	12	16	16	12	16	16
Optimization						
Batch size	256	256	256	256	256	1024
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	Muon
lr	0.0001	0.0001	0.0001	0.0001	0.0001	0.0002
(β_1, β_2)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.95)
EMA decay	0.9999	0.9999	0.9999	0.9999	0.9999	0.9995
Interpolants or Denoising						
α_t	$1-t$	$1-t$	$1-t$	-	-	$1-t$
σ_t	t	t	t	-	-	t
w_t	σ_t	σ_t	σ_t	-	-	σ_t
T	-	-	-	1000	1000	-
Training objective	v-prediction	v-prediction	v-prediction	noise-prediction	noise-prediction	v-prediction
Sampler	Euler-Maruyama	Euler-Maruyama	Euler-Maruyama	DDPM	DDPM	Heun
Sampling steps	250	250	250	250	250	125
CFG Scale	-	-	1.35 (if used)	-	-	1.45 (if used)
IG						
Auxiliary supervision position	4	8	8	4	8	8
IG Scale	2.3	2.3	1.4	2.3	2.3	1.4

F. Details of the 2D toy example

In this Section, we describe the construction of the 2D toy dataset used in the analysis of Section 4, as well as the associated model architecture, training setup, and sampling parameters.

Dataset. For each of the two classes $\mathbf{c}$, we model the fractal-like data distribution as a mixture of Gaussians $\mathcal{M}_\mathbf{c} = \big(\{\phi_i\}, \{\mu_i\}, \{\mathbf{\Sigma}_i\}\big)$, where ϕ_i, μ_i, and $\mathbf{\Sigma}_i$ represent the weight, mean, and 2×2 covariance matrix of each component i, respectively. This lets us calculate the ground truth scores and probability densities analytically and, consequently, to visualize them without making any additional assumptions. The probability density for a given class is given by

$$p_{\text{data}}(\mathbf{x} \mid \mathbf{c}) = \sum_{i \in \mathcal{M}_c} \phi_i \, \mathcal{N}(\mathbf{x}; \mu_i, \Sigma_i), \quad \text{where} \tag{1}$$

$$\mathcal{N}(\mathbf{x}; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^2 \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1} (\mathbf{x}-\mu)\right). \tag{2}$$

Applying heat diffusion to $p_{\text{data}}(\mathbf{x}|\mathbf{c})$, we obtain a sequence of increasingly smoothed densities $p(\mathbf{x}|\mathbf{c};\sigma)$ parameterized by noise level σ:

$$p(\mathbf{x} \mid \mathbf{c}; \sigma) = \sum_{i \in \mathcal{M}_c} \phi_i \, \mathcal{N}(\mathbf{x}; \mu_i, \Sigma^*_{i,\sigma}), \quad \text{where } \Sigma^*_{i,\sigma} = \Sigma_i + \sigma^2 \mathbf{I}. \tag{3}$$

The score function of $p(\mathbf{x}|\mathbf{c};\sigma)$ is then given by

$$\nabla_\mathbf{x} \log p(\mathbf{x} \mid \mathbf{c}; \sigma) = \frac{\sum_{i \in \mathcal{M}_c} \phi_i \, \mathcal{N}(\mathbf{x}; \mu_i, \Sigma^*_{i,\sigma}) \, (\Sigma^*_{i,\sigma})^{-1} (\mu_i - \mathbf{x})}{\sum_{i \in \mathcal{M}_c} \phi_i \, \mathcal{N}(\mathbf{x}; \mu_i, \Sigma^*_{i,\sigma})}. \tag{4}$$

We construct $\mathcal{M}_{\mathbf{c}}$ to represent a thin tree-like structure by starting with one main "branch" and recursively subdividing it into smaller ones. Each branch is represented by 8 anisotropic Gaussian components and the subdivision is performed 6 times, decaying ϕ after each subdivision and slightly randomizing the lengths and orientations of the two resulting sub-branches. This yields $127\times8 = 1016$ components per class and $1016\times2 = 2032$ components in total. We define the coordinate system so that the mean and standard deviation of p_{data}, marginalized over $\mathbf{c}$, are equal to 0 and $\sigma_{\text{data}} = 0.5$ along each axis, respectively, matching the recommendations by Karras et al. [25].

Models. We implement the denoiser models D as simple multi-layer perceptrons, utilizing the magnitude-preserving design principles from EDM2 [28]. Following Autoguidance [26], we design the model interface so that for a given noisy sample, each model outputs a single scalar representing the logarithm of the corresponding unnormalized probability density, as opposed to directly outputting the denoised sample or the score vector. Concretely, let us denote the output of a given model by $G_\theta(\mathbf{x}; \sigma, \mathbf{c})$. The corresponding normalized probability density is then given by

$$p_\theta(\mathbf{x} \mid \mathbf{c}; \sigma) = \exp\big(G_\theta(\mathbf{x}; \sigma, \mathbf{c})\big) \Big/ \int \exp\big(G_\theta(\mathbf{x}; \sigma, \mathbf{c})\big)\, d\mathbf{x}. \tag{5}$$

By virtue of defining G_θ this way, we can derive the score vector, and by extension, the denoised sample, from G_θ through automatic differentiation:

$$\nabla_x \log p_\theta(x|c; \sigma) = \nabla_x G_\theta(x; \sigma, c) \tag{6}$$

$$D_\theta(x; \sigma, c) = x + \sigma^2 \nabla_x G_\theta(x; \sigma, c). \tag{7}$$

Besides Equation 6, according to the description of Autoguidance, trying out the alternative formulations where the model outputs the score vector or the denoised sample directly is qualitatively more or less identical.

To connect the above definition of G_θ to the raw network layers, we apply preconditioning using the same general principles as in EDM [24]. Denoting the function represented by the raw network layers as F_θ, we define G_θ as

$$G_\theta(\mathbf{x}; \sigma, c) = -\frac{1}{2}\|\mathbf{x}\|_2^2 - \frac{g_\theta}{\sigma n} \sum_{i=1}^{n} F_{\theta,i}\left(\mathbf{x}^*; \frac{1}{4}\log\sigma, c\right)^2, \quad \text{where} \quad \mathbf{x}^* = \frac{\mathbf{x}}{\sqrt{\sigma^2 + \sigma_{\text{data}}^2}} \tag{8}$$

and the sum is taken over the n output features of F_θ. We scale the output of F_θ by a learned scaling factor g_θ that we initialize to zero. The goal of Equation 8 is to satisfy the following three requirements:

- The input of F_θ should have zero mean and unit magnitude. This is achieved through the division by $\sqrt{\sigma^2 + \sigma_{\text{data}}^2}$.
- After initialization, G_θ should represent the best possible first-order approximation of the correct solution. This is achieved through the $-\frac{1}{2}\|\mathbf{x}^*\|_2^2$ term, as well as the fact that $g_\theta = 0$ after initialization.
- After training, $\sqrt{g_\theta} \cdot F_\theta$ should have approximately unit magnitude. This is achieved through the division by σn.

In practice, we use an MLP with one input layer and four hidden layers, interspersed with SiLU [15] activation functions and implemented using the magnitude-preserving primitives from EDM2 [27]. Additionally, we defined an output layer after the first hidden layer to produce a weaker version of the output. The input is a 4-dimensional vector $\left[\mathbf{x}_x^*; \mathbf{x}_y^*; \frac{1}{4}\log\sigma; 1\right]$ and the output of each hidden layer has n features, where $n = 64$.

Training. Given that we have the exact score function of the ground truth distribution readily available (Equation 3), we train the models using exact score matching [21] for simplicity and increased robustness. We thus define the loss function and the additional supervision loss as

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\sigma\sim p_{\text{train}}, \mathbf{x}\sim p(\mathbf{x};\sigma)} \sigma^2 \left\|\nabla_{\mathbf{x}} \log p_{\theta_i}(\mathbf{x}; \sigma) - \nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma)\right\|_2^2, \tag{9}$$

$$\mathcal{L}(\theta_d) = \mathbb{E}_{\sigma\sim p_{\text{train}}, \mathbf{x}\sim p(\mathbf{x};\sigma)} \sigma^2 \left\|\nabla_{\mathbf{x}} \log p_{\theta_d}(\mathbf{x}; \sigma) - \nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma)\right\|_2^2, \tag{10}$$

where $\sigma \sim p_{\text{train}}$ is realized as $\log(\sigma) \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}})$ [25] and θ_i and θ_d represent the intermediate layer and the deep layer network respectively. In particular, for the training acceleration method depicted in Figure 4, our loss function is

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\sigma\sim p_{\text{train}}, \mathbf{x}\sim p(\mathbf{x};\sigma)} \sigma^2 \left\|\nabla_{\mathbf{x}} \log p_{\theta_i}(\mathbf{x}; \sigma) - \nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma)\right\|_2^2, \tag{11}$$

$$\mathcal{L}(\theta_d) = \mathbb{E}_{\sigma\sim p_{\text{train}},\mathbf{x}\sim p(\mathbf{x};\sigma)}\sigma^2 \left\|\nabla_{\mathbf{x}} \log p_{\theta_d}(\mathbf{x};\sigma) - (\nabla_{\mathbf{x}} \log p(\mathbf{x};\sigma) + 0.5 * (\nabla_{\mathbf{x}} \log p_{\theta_d}(\mathbf{x};\sigma) - \nabla_{\mathbf{x}} \log p_{\theta_i}(\mathbf{x};\sigma)))\right\|_2^2, \tag{12}$$

The final training loss is $\mathcal{L} = \mathcal{L}(\theta_d) + 0.5 * \mathcal{L}(\theta_i)$ The more commonly used denoising score matching do not show any significant differences in the model behavior or training dynamics.

We train these models for 4096 iterations using a batch size of 4096 samples. We set $P_{\text{mean}} = -2.3$ and $P_{\text{std}} = 1.5$, and use $\alpha_{\text{ref}}/\sqrt{\max(t/t_{\text{ref}}, 1)}$ learning rate decay schedule with $\alpha_{\text{ref}} = 0.01$ and $t_{\text{ref}} = 512$ iterations, along with a power function EMA profile [27] with $\sigma_{\text{rel}} = 0.010$. Overall, the setup is robust with respect to the hyperparameters; the phenomena illustrated in Figures 3 and 4 remain unchanged across a wide range of parameter choices.

Sampling. We use the standard EDM sampler [25] with $N = 32$ Heun steps (NFE $= 63$), $\sigma_{\text{min}} = 0.002$, $\sigma_{\text{max}} = 5$, and $\rho = 7$. We chose the values of N and σ_{max} to be much higher than what is actually needed for this dataset in order to avoid potential discretization errors from affecting our conclusions. In Figure 3, we set $w = 2.5$ for CFG, $w = 2$ for autoguidance and $w = 2$ for internal guidance (IG), and $w_1 = 1$ and $w_2 = 1.5$ for IG + CFG. In Figure 4, we set $w = 2$ for internal guidance.

G. More Discussion on Related Work

Denoising transformers. Many recent works have tried to use transformer backbones for diffusion or flow-based model training. First, several works like U-ViT [1], MDT [11], and DiffiT [14] how transformer-based backbones with skip connections can be an effective backbone for training diffusion models. Intriguingly, DiT and SiT show a pure transformer architecture can be a scalable architecture for training diffusion-based models. Based on these improvements, Stable diffusion 3 [7], FLUX 1 [33] and Nano Banana [13] show pure transformers can be scaled up for challenging text-to-image generation, and Sora, CogvideoX [58] and Wan [54] demonstrate their success in text-to-video generation. Our work analyzes and improves the training of Diffusion Transformer architecture based on a simple auxiliary supervision to the early layers.

Sampling guidance in diffusion models. Achieving better generation results over diffusion models remains challenging yet essential. Early approaches, such as classifier guidance [6], introduce control by incorporating classifier gradients into the sampling process. However, this method requires separately trained classifiers, making it less flexible and computationally demanding. To overcome these limitations, classifier free guidance (CFG) [17] is proposed, enabling guidance without the need for an external classifier. Instead, CFG trains conditional and unconditional models simultaneously and interpolates between their outputs during sampling. Subsequently, several studies [2, 4, 8, 10, 32, 46, 56] have reduced the downsides of CFG by making the guidance weight noise level-dependent. Another research direction separates the class guidance part from the CFG and only uses a weaker version of the current model for guidance [3, 18–20, 26], which also bears conceptual similarity to contrastive decoding [38] used in large language models to reduce the repetitiveness of generations. Our approach does not require any additional sampling steps. It can be applied to any deep diffusion Transformer network in a plug-and-play manner while achieving better generation results.

Efficient training of generative models. The issue of how to accelerate the convergence of generative models during the training process has also received significant attention. Previous studies have explored architectural optimization [29, 42, 52], improved flow-based theories [9, 39], optimized data [53, 61] to accelerate convergence. Meanwhile, there have been several approaches in generative adversarial network [12] that try to accelerate training with better convergence using pretrained visual encoder [23, 30, 48, 49]. Another line of work tries to exploit the pretrained visual encoders for improving diffusion model training from scratch [36, 43], usually by training two diffusion models where one model generates the pretrained representations and the other model generates the target data conditioned on the generated representation. More recently, representation learning has been introduced into generative model training to further enhance training efficiency [34, 35, 57, 59, 60, 62]. Our method not only alleviate the problem of gradient vanishing during training, but also can be directly applied in the sampling stage to achieve better generation results on models with less training consumption.

H. More Qualitative Results

Below we show some uncurated generation results on ImageNet 256×256 from the SiT-XL+IG (1.8). We use classifier-free guidance with w = 2.5.

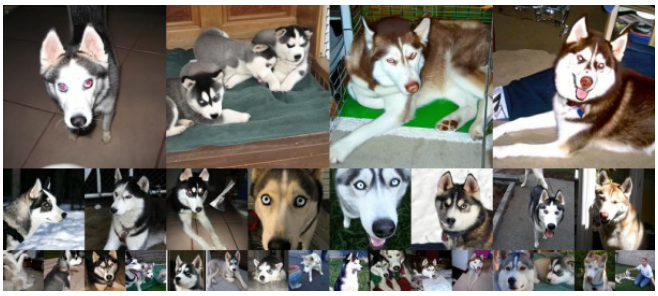
Class 022: bald eagle, American eagle, Haliaeetus leucocephalus

Class 024: great grey owl, great gray owl, Strix nebulosa

Class 033: loggerhead, loggerhead turtle, Caretta caretta

Class 089: sulphur-crested cockatoo, Kakatoe galerita, Cacatua galerita

Class 207: golden retriever



Class 250: Siberian husky

Class 264: Cardigan, Cardigan Welsh corgi

Class 288: leopard, Panthera pardus

Figure 2. Uncurated visualization results of LightningDiT-XL/1 + IG (1.8) use CFG with $w = 2.5$.

Class 309: bee

Class 388: giant panda, panda, panda bear, coon bear, Ailuropoda melanoleuca

Class 417: balloon

Class 425: barn

Class 429: baseball

Class 511: convertible

Class 661: Model T

Class 780: schooner

Figure 3. Uncurated visualization results of LightningDiT-XL/1 + IG (1.8) use CFG with $w = 2.5$.

Class 873: triumphal arch

Class 928: ice cream, icecream

Class 930: French loaf

Class 933: cheeseburger

Class 947: mushroom

Class 971: bubble

Class 975: lakeside, lakeshore

Class 980: volcano

Figure 4. Uncurated visualization results of LightningDiT-XL/1 + IG (1.8) use CFG with $w = 2.5$.

References

[1] Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22669–22679, 2023. 6

[2] Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*, 2023. 6

[3] Chubin Chen, Jiashu Zhu, Xiaokun Feng, Nisha Huang, Meiqi Wu, Fangyuan Mao, Jiahong Wu, Xiangxiang Chu, and Xiu Li. s^2-guidance: Stochastic self guidance for training-free enhancement of diffusion models. *arXiv preprint arXiv:2508.12880*, 2025. 6

[4] Hyungjin Chung, Jeongsol Kim, Geon Yeong Park, Hyelin Nam, and Jong Chul Ye. Cfg++: Manifold-constrained classifier free guidance for diffusion models. *arXiv preprint arXiv:2406.08070*, 2024. 6

[5] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023. 2

[6] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 1, 6

[7] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024. 6

[8] Weichen Fan, Amber Yijia Zheng, Raymond A Yeh, and Ziwei Liu. Cfg-zero*: Improved classifier-free guidance for flow matching models. *arXiv preprint arXiv:2503.18886*, 2025. 6

[9] Johannes S Fischer, Ming Gui, Pingchuan Ma, Nick Stracke, Stefan A Baumann, and Björn Ommer. Boosting latent diffusion with flow matching. *arXiv preprint arXiv:2312.07360*, 2023. 6

[10] Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Masked diffusion transformer is a strong image synthesizer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 23164–23173, 2023. 6

[11] Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Mdtv2: Masked diffusion transformer is a strong image synthesizer. *arXiv preprint arXiv:2303.14389*, 2023. 6

[12] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 6

[13] Google. Nano banana:, 2025. 6

[14] Ali Hatamizadeh, Jiaming Song, Guilin Liu, Jan Kautz, and Arash Vahdat. Diffit: Diffusion vision transformers for image generation. In *European Conference on Computer Vision*, pages 37–55. Springer, 2024. 6

[15] D Hendrycks. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 5

[16] 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. 1

[17] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 2, 6

[18] Susung Hong. Smoothed energy guidance: Guiding diffusion models with reduced energy curvature of attention. *Advances in Neural Information Processing Systems*, 37:66743–66772, 2024. 6

[19] Susung Hong, Gyuseong Lee, Wooseok Jang, and Seungryong Kim. Improving sample quality of diffusion models using self-attention guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7462–7471, 2023.

[20] Junha Hyung, Kinam Kim, Susung Hong, Min-Jung Kim, and Jaegul Choo. Spatiotemporal skip guidance for enhanced video diffusion sampling. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 11006–11015, 2025. 6

[21] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005. 5

[22] 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. 2, 3

[23] 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. 6

[24] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Proc. NeurIPS*, 2022. 5

[25] 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. 5, 6

[26] Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. *Advances in Neural Information Processing Systems*, 37:52996–53021, 2024. 2, 5, 6

[27] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *Proc. CVPR*, 2024. 5, 6

[28] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24174–24184, 2024. 2, 5

[29] Felix Krause, Timy Phan, Ming Gui, Stefan Andreas Baumann, Vincent Tao Hu, and Björn Ommer. Tread: Token routing for efficient architecture-agnostic diffusion training. *arXiv preprint arXiv:2501.04765*, 2025. 6

[30] Nupur Kumari, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Ensembling off-the-shelf models for gan training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10651–10662, 2022. 6

[31] 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. 1

[32] Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models. *Advances in Neural Information Processing Systems*, 37: 122458–122483, 2024. 6

[33] Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025. 6

[34] Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. Repa-e: Unlocking vae for end-to-end tuning with latent diffusion transformers. *arXiv preprint arXiv:2504.10483*, 2025. 1, 6

[35] Tianhong Li, Dina Katabi, and Kaiming He. Return of unconditional generation: A self-supervised representation generation method. In *Advances in Neural Information Processing Systems*, pages 125441–125468. Curran Associates, Inc., 2024. 6

[36] Tianhong Li, Dina Katabi, and Kaiming He. Return of unconditional generation: A self-supervised representation generation method. *Advances in Neural Information Processing Systems*, 37:125441–125468, 2024. 6

[37] 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. 1

[38] Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori B Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 12286–12312, 2023. 6

[39] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 6

[40] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W Battaglia. Generating images with sparse representations. *arXiv preprint arXiv:2103.03841*, 2021. 1

[41] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 3

[42] 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. 6

[43] Pablo Pernias, Dominic Rampas, Mats L Richter, Christopher J Pal, and Marc Aubreville. Würstchen: An efficient architecture for large-scale text-to-image diffusion models. *arXiv preprint arXiv:2306.00637*, 2023. 6

[44] Sucheng Ren, Qihang Yu, Ju He, Xiaohui Shen, Alan Yuille, and Liang-Chieh Chen. Beyond next-token: Next-x prediction for autoregressive visual generation. *arXiv preprint arXiv:2502.20388*, 2025. 1

[45] 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. 2

[46] Seyedmorteza Sadat, Jakob Buhmann, Derek Bradley, Otmar Hilliges, and Romann M Weber. Cads: Unleashing the diversity of diffusion models through condition-annealed sampling. *arXiv preprint arXiv:2310.17347*, 2023. 6

[47] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016. 1

[48] Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster. *Advances in Neural Information Processing Systems*, 34:17480–17492, 2021. 6

[49] 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. 6

[50] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 1

[51] 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. 1

[52] Yuchuan Tian, Jing Han, Chengcheng Wang, Yuchen Liang, Chao Xu, and Hanting Chen. Dic: Rethinking conv3x3 designs in diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2469–2478, 2025. 6

[53] Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023. 6

[54] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 6

[55] Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. Ddt: Decoupled diffusion transformer. *arXiv preprint arXiv:2504.05741*, 2025. 1

[56] Xi Wang, Nicolas Dufour, Nefeli Andreou, Marie-Paule Cani, Victoria Fernández Abrevaya, David Picard, and Vicky Kalogeiton. Analysis of classifier-free guidance weight schedulers. *arXiv preprint arXiv:2404.13040*, 2024. 6

[57] Ge Wu, Shen Zhang, Ruijing Shi, Shanghua Gao, Zhenyuan Chen, Lei Wang, Zhaowei Chen, Hongcheng Gao, Yao Tang, Jian Yang, et al. Representation entanglement for generation: Training diffusion transformers is much easier than you think. *arXiv preprint arXiv:2507.01467*, 2025. 6

[58] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024. 6

[59] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15703–15712, 2025. 1, 2, 6

[60] 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. *arXiv preprint arXiv:2410.06940*, 2024. 1, 2, 6

[61] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 6

[62] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. *arXiv preprint arXiv:2510.11690*, 2025. 1, 6