Supplementary Materials

A. Implementation Details

Our implementation closely follows the public codebases of DiT [16] and SiT [15]. Our configurations are summarized in Tab. 1. We describe the details as follows.

Time distribution. Following [3], during training, we adopt the logit-normal distribution over t [3]: $\text{logit}(t) \sim \mathcal{N}(\mu, \sigma^2)$. Specifically, we sample $s \sim \mathcal{N}(\mu, \sigma^2)$ and let $t{=}\text{sigmoid}(s)$. The hyper-parameter μ determines the noise level (Tab. 3, main paper). By default we set $\mu = -0.8$ on ImageNet at resolution 256 (or 512, 1024), and fix σ as 0.8.

ImageNet 512×512 and 1024×1024. We adopt **JiT/32** (*i.e.*, a patch size of 32) on ImageNet 512×512. The model leads to a sequence of 256 = 16×16 patches, the same as JiT/16 on ImageNet 256×256. As such, JiT/32 only differs from JiT/16 in the input/output patch dimension, increasing from 768-d to 3072-d per patch; all other computations and costs are exactly the same.

To reuse the exact same recipe from ImageNet 256×256, for 512×512 images we rescale the magnitude of the noise $\boldsymbol{\epsilon}$ by 2×: that is, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 2^2\mathbf{I})$. This simple modification approximately maintains the signal-to-noise ratio (SNR) between the 256×256 and 512×512 resolutions [8, 1, 9]. No other changes to the ImageNet 256×256 configuration are required or applied.

For ImageNet 1024×1024, we use the model JiT/64 and scale the noise $\boldsymbol{\epsilon}$ by 4×. No other change is needed.

In-context class conditioning. Standard DiT [16] performs class conditioning through adaLN-Zero. In Tab. 4, main paper, we further explore in-context class-conditioning.

Specifically, following ViT [2], one can prepend a class token to the sequence of patches. This is referred to as "in-context conditioning" in DiT [16]. Note that we use in-context conditioning jointly with the default adaLN-Zero conditioning, unlike DiT. In addition, following MAR [14], we further prepend multiple such tokens to the sequence. These tokens are repeated instances of the same class token, with different positional embeddings added. We prepend 32 such tokens. Moreover, rather than prepending these tokens to the Transformer's input, we find that prepending them at later blocks can be beneficial (see "in-context start block" in Tab. 1). Tab. 4, main paper shows that our implementation of in-context conditioning improves FID by ∼1.2.

Dropout and early stop. We apply dropout [19] in JiT-H and G to mitigate the risk of overfitting. Specifically, we apply dropout to the middle half of the Transformer blocks. For Transformer blocks with dropout, we apply it to both the attention block and the MLP block.

As the G-size models still tend to overfit under our current dropout setting, we apply early stopping when the monitored FID begins to degrade. This occurs at around 320 epochs for both JiT-G/16 and JiT-G/32.

	JiT-B	JiT-L	JiT-H	JiT-G
architecture				
depth	12	24	32	40
hidden dim	768	1024	1280	1664
heads	12	16	16	16
image size	256 (other settings: 512, or 1024)			
patch size	`image_size` / 16			
bottleneck	128 (B/L), 256 (H/G)			
dropout	0 (B/L), 0.2 (H/G)			
in-context class tokens	32 (if used)			
in-context start block	4	8	10	10
training				
epochs	200 (ablation), 600			
warmup epochs [4]	5			
optimizer	Adam [11], $\beta_1, \beta_2 = 0.9, 0.95$			
batch size	1024			
learning rate	2e-4			
learning rate schedule	constant			
weight decay	0			
ema decay	{0.9996, 0.9998, 0.9999}			
time sampler	$\text{logit}(t) \sim \mathcal{N}(\mu, \sigma^2)$, $\mu = -0.8$, $\sigma = 0.8$			
noise scale	$1.0 \times$ `image_size` / 256			
clip of $(1-t)$ in division	0.05			
class token drop (for CFG)	0.1			
sampling				
ODE solver	Heun [5]			
ODE steps	50			
time steps	linear in [0.0, 1.0]			
CFG scale sweep range [7]	[1.0, 4.0]			
CFG interval [13]	[0.1, 1] (if used)			

Table 1. **Configurations of experiments.**

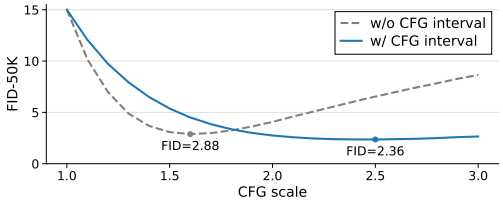


Figure 1. The influence of CFG, without and with CFG interval (for JiT-L/16 on ImageNet 256×256, 600 epochs).

EMA and CFG. Our study covers a wide range of configurations, including variations in loss and prediction spaces, model sizes, and architectural components. The optimal values of the CFG scale [7] and EMA (exponential moving average) decay vary from case to case, and fixing them may lead to incomplete or misleading observations. Since maintaining these hyperparameter configurations is relatively inexpensive, we strive to adopt the optimal values.

Specifically, for the CFG scale ω [7], we determine the optimal value by searching over a range of candidate scales at inference time, as is common practice in existing work. For EMA decays, we maintain multiple copies of the moving-averaged parameters during training, which introduces a negligible computational overhead. To avoid memory overhead, different EMA copies can be stored on separate devices (*e.g.*, GPUs). As such, both the CFG scale and EMA decay are essentially inference-time decisions.

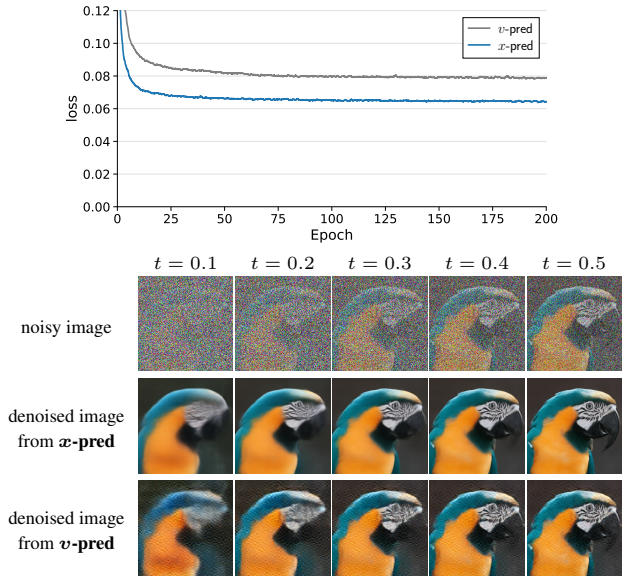


Figure 2. **(Top)**: **Training loss** of $\boldsymbol{x}$- and $\boldsymbol{v}$-prediction, using the same loss space of $\boldsymbol{v}$-loss (Tab. 2(a), third row, main paper). We plot the loss averaged per pixel per channel. **(Bottom)**: **Denoised images** from $\boldsymbol{x}$- and $\boldsymbol{v}$-prediction, where $\boldsymbol{v}$-prediction's denoised output is visualized according to Tab. 1(c)(1), main paper. The denoised image from $\boldsymbol{v}$-prediction has noticeable artifacts, as reflected by the higher loss.

Our CFG scale candidates range from 1.0 to 4.0 with a step size of 0.1. The influence of CFG is shown in Fig. 1 for a JiT-L/16 model. Our EMA decay candidates are 0.9996, 0.9998, and 0.9999, evaluated with a batch size of 1024. For each model (including any one in the ablation), we search for the optimal setting using 8K samples and then apply it to evaluate 50K samples.

Evaluation. Following common pratice, we evaluate the FID [6] against the ImageNet training set. We evaluate FID on 50K generated images, with 50 samples for each of the 1000 ImageNet classes. We evaluate the Inception Score (IS) [18] on the same 50K images.

B. Additional Experiments

B.1. Training Loss and Denoised Images

In Tab. 2(a), main paper, the failure of $\boldsymbol{\epsilon}$-/$\boldsymbol{v}$-prediction is caused by the inherent incapability of predicting a high-dimensional output from a limited-capacity network. This failure can be seen from the training loss curves.

In Fig. 2 (top), we compare the training loss curves under the same $\boldsymbol{v}$-loss, defined as $\mathcal{L}=\mathbb{E}\|\boldsymbol{v}_\theta(\boldsymbol{z}_t,t)-\boldsymbol{v}\|^2$, using $\boldsymbol{v}$-prediction (*i.e.*, $\boldsymbol{v}_\theta=\texttt{net}_\theta$) versus $\boldsymbol{x}$-prediction (*i.e.*, $\boldsymbol{v}_\theta=(\texttt{net}_\theta-\boldsymbol{z}_t)/(1-t)$). Since the loss is computed in the same space and only the parameterization differs, comparing the loss values is legitimate.

Fig. 2 (top) shows that $\boldsymbol{v}$-prediction yields substantially higher loss (about 25%) than $\boldsymbol{x}$-prediction, even though $\boldsymbol{v}$-prediction appears to be the "native" parameterization for the $\boldsymbol{v}$-loss. This comparison indicates that the task of $\boldsymbol{x}$-prediction is inherently *easier*, as the data lie on a low-dimensional manifold. We also observe that $\boldsymbol{\epsilon}$-prediction (not shown here) has about $3\times$ higher loss and is unstable, which may explain its failure in Tab. 2(a), main paper.

Fig. 2 (bottom) compares the *denoised* images corresponding to the two training curves. For $\boldsymbol{x}$-prediction, the denoised image is simply $\boldsymbol{x}_\theta=\texttt{net}_\theta$; for $\boldsymbol{v}$-prediction, the denoised image is $\boldsymbol{x}_\theta=(1-t)\texttt{net}_\theta+\boldsymbol{z}_t$ with $\boldsymbol{v}_\theta=\texttt{net}_\theta$ (see Tab. 1(c)(1), main paper). The *higher* loss of $\boldsymbol{v}$-prediction in Fig. 2 (top) can be reflected by its noticeable *artifacts* in Fig. 2 (bottom).

Note that the artifact in Fig. 2 (bottom) is that of a *single* denoising step. In the generation process, this error can accumulate in the multi-step ODE solver, which leads to the catastrophic failure in Tab. 2(a), main paper.

B.2. Pre-conditioner

In EDM [10], an extra "pre-conditioner" is applied to wrap the direct output of the network. Using the notation of our paper, the pre-conditioner formulation can be written as: $\boldsymbol{x}_\theta(\boldsymbol{z}_t,t)=c_{\text{skip}}\cdot\boldsymbol{z}_t+c_{\text{out}}\cdot\texttt{net}_\theta(\boldsymbol{z}_t,t)$, where c_{skip} and c_{out} are pre-defined coefficients. This equation suggests that *unless* $c_{\text{skip}}\equiv 0$, the network output in a pre-conditioner must *not* perform $\boldsymbol{x}$-prediction. And according to the manifold assumption, this formulation should not remedy the issue we consider, as we examine next.

Formulation of pre-conditioners. Given the definition of pre-conditioner, we can rewrite the "pre-conditioned $\boldsymbol{x}$-prediction" as:

$$\begin{cases}\boldsymbol{x}_\theta = c_{\text{skip}}\cdot\boldsymbol{z}_t + c_{\text{out}}\cdot\texttt{net}_\theta\\ \boldsymbol{\epsilon}_\theta = (\boldsymbol{z}_t-t\boldsymbol{x}_\theta)/(1-t)\\ \boldsymbol{v}_\theta = (\boldsymbol{x}_\theta-\boldsymbol{z}_t)/(1-t)\end{cases}\tag{1}$$

Accordingly, the objective in Eq. (6) ($\boldsymbol{v}$-loss), main paper is written as:

$$\mathcal{L}=\mathbb{E}_{t,\boldsymbol{x},\boldsymbol{\epsilon}}\Big\|\boldsymbol{v}_\theta(\boldsymbol{z}_t,t)-\boldsymbol{v}\Big\|^2,\tag{2}$$
$$\text{where:}\quad \boldsymbol{v}_\theta(\boldsymbol{z}_t,t)=(\boldsymbol{x}_\theta(\boldsymbol{z}_t,t)-\boldsymbol{z}_t)/(1-t)$$
$$\text{and}\quad \boldsymbol{x}_\theta(\boldsymbol{z}_t,t)=c_{\text{skip}}\cdot\boldsymbol{z}_t+c_{\text{out}}\cdot\texttt{net}_\theta(\boldsymbol{z}_t,t).$$

Comparing with Eq. (6), main paper, the only difference is in how we compute $\boldsymbol{x}_\theta$ from the network.

As EDM [10] uses a variance-exploding schedule (that is, $\boldsymbol{z}_t=\boldsymbol{x}+\sigma_t\boldsymbol{\epsilon}$) and we use a (roughly) variance-preserving version (that is, $\boldsymbol{z}_t=t\boldsymbol{x}+(1-t)\boldsymbol{\epsilon}$), a fully equivalent conversion is impossible. To have a pre-conditioner in our case, we rewrite our version as $\frac{1}{t}\boldsymbol{z}_t=\boldsymbol{x}+\frac{1-t}{t}\boldsymbol{\epsilon}$. As such, we set $\sigma_t=\frac{1-t}{t}$, which is the noise added to unscaled images, similar to EDM's. With this, we can rewrite the co-

		pre-conditioned predictions	
	$\boldsymbol{x}$**-pred**	EDM-style	linear
	$c_{\text{skip}}=0$ $c_{\text{out}}=1$	$c_{\text{skip}}=\frac{1}{t}\frac{\sigma_{\text{data}}^2}{\sigma_{\text{data}}^2+\sigma_t^2}$ $c_{\text{out}}=\frac{\sigma_{\text{data}}\sigma_t}{\sqrt{\sigma_{\text{data}}^2+\sigma_t^2}}$	$c_{\text{skip}}=t$ $c_{\text{out}}=1-t$
$\boldsymbol{x}$**-loss**	10.14	28.94	39.50
ϵ**-loss**	10.45	72.05	67.56
$\boldsymbol{v}$**-loss**	8.62	35.49	46.25

Table 2. **Comparisons with pre-conditioners** (FID-50K, ImageNet 256). The settings are the same as Tab. 2(a), main paper.

FID-50K	**JiT-B/16**	**JiT-L/16**
$\boldsymbol{v}$-loss only	4.37	2.79
w/ cls loss	**4.14**	**2.50**

Table 3. **Exploration: additional classification loss**. We do ***not*** use this loss in any other experiments. Settings: ImageNet 256×256, 200-ep, with CFG interval.

efficients defined by EDM [10] as: $c_{\text{skip}}=\frac{1}{t}\frac{\sigma_{\text{data}}^2}{\sigma_{\text{data}}^2+\sigma_t^2}$ and $c_{\text{out}}=\frac{\sigma_{\text{data}}\sigma_t}{\sqrt{\sigma_{\text{data}}^2+\sigma_t^2}}$ where σ_{data} is the data standard deviation set as 0.5 [10]. We choose $c_{\text{in}}=t\ (=\frac{1}{\sigma_t+1})$, so that the direct input to the network (*i.e.*, $c_{\text{in}}\cdot\frac{1}{t}z_t$) is still z_t. It can be shown that *only* when $t\to 0$, we have: $\sigma_t\to+\infty$, $c_{\text{skip}}\to 0$, $c_{\text{out}}\to 1$, which approaches $\boldsymbol{x}$-prediction. We also consider a simpler *linear* pre-conditioner: $c_{\text{skip}}=t$ and $c_{\text{out}}=1-t$, which also performs $\boldsymbol{x}$-prediction *only* when $t=0$.

Results of pre-conditioners. Tab. 2 shows that the pre-conditioned versions all fail catastrophically, suggesting that deviating from $\boldsymbol{x}$-prediction is not desired in high-dimensional spaces. Interestingly, the pre-conditioned versions are much better than $\boldsymbol{\epsilon}$-/$\boldsymbol{v}$-prediction in Tab. 2(a), main paper. We hypothesize that this is because they are more similar to $\boldsymbol{x}$-prediction when $t\to 0$, which alleviates this issue.

B.3. Exploration: Classification Loss

Our paper focuses on a minimalist design, and we intentionally do *not* use any extra loss. However, we note that latent-based methods [17] typically rely on tokenizers trained with *adversarial* and *perceptual* losses, and thus their generation process is not purely driven by diffusion. Next, we discuss a simple extension on our pixel-based models with an additional classification loss.

Formally, we attach a classifier head after a specific Transformer block (the 4th in JiT-B and the 8th in JiT-L). The classifier consists of global average pooling followed by a linear layer, and a cross-entropy loss is applied for the 1000-class ImageNet classification task. This classification loss $\mathcal{L}_{\text{cls}}$ is scaled by a weight λ (*e.g.*, 100) and added to the ℓ_2-based (*i.e.*, element-wise sum of squared errors) regression loss. To prevent label leakage, we disable class conditioning for all layers before the classifier head. We note that this modification remains minimal and does not rely on any pre-trained classifier, unlike the perceptual loss [20]. This minor modification leads to a decent improvement, as shown in Tab. 3. This exploration suggests further potential for combining our simple method with additional loss terms, which we leave for future work.

Despite the improvement, we do ***not*** use this or any additional loss in the other experiments presented in this paper.

B.4. Cross-resolution Generation

A model trained at one resolution can be applied to another by simply downsampling or upsampling the generated images. We refer to this as *cross-resolution* generation. In our setup, JiT/16 at 256 and JiT/32 at 512 have comparable parameters and compute, making their cross-resolution comparison meaningful. The results are in Tab. 4

	FID@256		FID@512
JiT-G/16@256	1.82	**JiT-G/16**@256, ↑512	2.45
JiT-G/32@512, ↓256	1.84	**JiT-G/32**@512	1.78

Table 4. **Cross-resolution Generation** (noted in gray), using JiT-G/16 trained at resolution 256 and JiT-G/32 trained at 512, followed by upsampling (↑) or downsampling (↓). All entries have similar parameters and flops (see Tab. 7 and 8, main paper).

Downsampling the images generated by the 512 model to 256 resolution yields a decent FID@256 of 1.84. This result remains competitive when compared with the 256-resolution expert (FID@256 of 1.82), while maintaining a similar computational cost and gaining the additional ability to generate at 512 resolution.

On the other hand, *upsampling* the images generated by the 256 model to 512 resolution results in a noticeably worse FID@512 of 2.45, compared with the 512-resolution expert's FID@512 of 1.78. This degradation is caused by the loss of higher-frequency details due to upsampling.

B.5. Additional Metrics

For completeness, we report precision and recall [12] on ImageNet 256×256 in Tab. 5, compared with the commonly used baselines of DiT and SiT, and the latest RAE:

	FID↓	IS↑	Prec↑	Rec↑
DiT-XL/2 [16]	2.27	278.2	0.83	0.57
SiT-XL/2 [15]	2.06	277.5	0.82	0.59
RAE [21], DiT$^{\text{DH}}$-XL/2	1.13	262.6	0.78	0.67
JiT-B/16	3.66	275.1	0.82	0.50
JiT-L/16	2.36	298.5	0.80	0.59
JiT-H/16	1.86	303.4	0.78	0.62
JiT-G/16	1.82	292.6	0.79	0.62

Table 5. Precision and recall on ImageNet 256×256.

C. Qualitative Results

In Fig. 3 to 6, we provide additional *uncurated* examples on ImageNet 256×256.

class 012: house finch, linnet, Carpodacus mexicanus

class 014: indigo bunting, indigo finch, indigo bird, Passerina cyanea

class 042: agama

class 081: ptarmigan

class 107: jellyfish

class 108: sea anemone, anemone

class 110: flatworm, platyhelminth

class 117: chambered nautilus, pearly nautilus, nautilus

class 130: flamingo

class 279: Arctic fox, white fox, Alopex lagopus

Figure 3. *Uncurated* samples on ImageNet 256×256 using JiT-G/16 conditioned on the specified classes. Unlike the common practice of visualizing with a higher CFG, here we show images using the CFG value (2.2) that achieves the reported FID of 1.82.

class 288: leopard, Panthera pardus

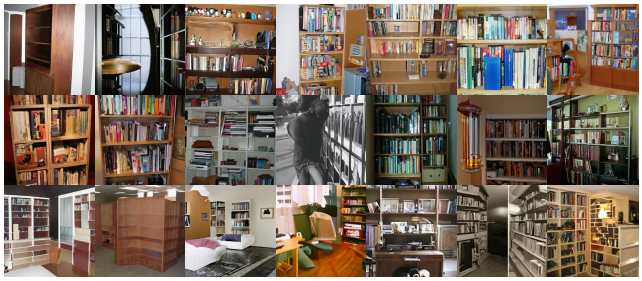
class 309: bee

class 349: bighorn, bighorn sheep, cimarron, Rocky Mountain bighorn

class 397: puffer, pufferfish, blowfish, globefish

class 425: barn

class 448: birdhouse



class 453: bookcase

class 458: brass, memorial tablet, plaque

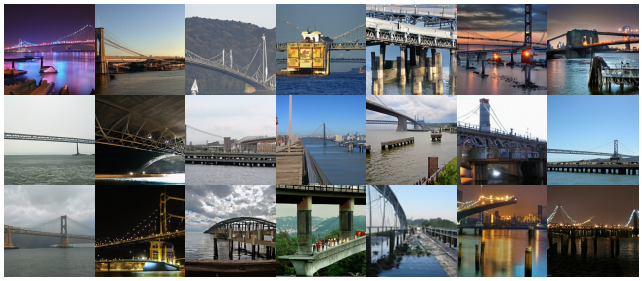
class 495: china cabinet, china closet

class 500: cliff dwelling

Figure 4. *Uncurated* samples on ImageNet 256×256 using JiT-G/16 conditioned on the specified classes. Unlike the common practice of visualizing with a higher CFG, here we show images using the CFG value (2.2) that achieves the reported FID of 1.82.

class 658: mitten

class 661: Model T



class 718: pier

class 724: pirate, pirate ship

class 725: pitcher, ewer

class 757: recreational vehicle, RV, R.V.

class 779: school bus

class 780: schooner

class 829: streetcar, tram, tramcar, trolley, trolley car

class 853: thatch, thatched roof

Figure 5. *Uncurated* samples on ImageNet 256×256 using JiT-G/16 conditioned on the specified classes. Unlike the common practice of visualizing with a higher CFG, here we show images using the CFG value (2.2) that achieves the reported FID of 1.82.

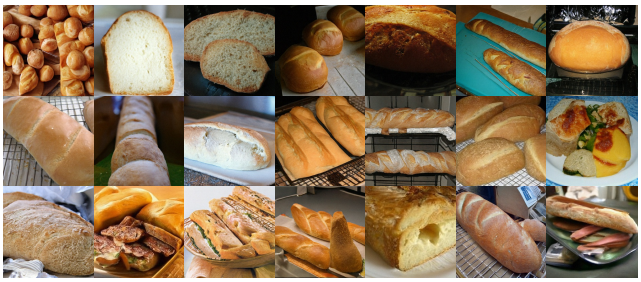
class 873: triumphal arch

class 900: water tower

class 911: wool, woolen, woollen

class 913: wreck

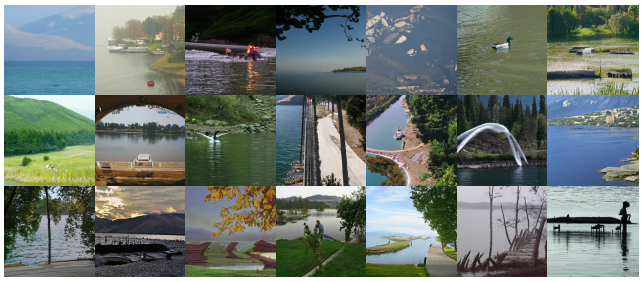
class 927: trifle



class 930: French loaf

class 946: cardoon

class 947: mushroom



class 975: lakeside, lakeshore

class 989: hip, rose hip, rosehip

Figure 6. *Uncurated* samples on ImageNet 256×256 using JiT-G/16 conditioned on the specified classes. Unlike the common practice of visualizing with a higher CFG, here we show images using the CFG value (2.2) that achieves the reported FID of 1.82.

References

[1] Ting Chen. On the importance of noise scheduling for diffusion models. *arXiv:2301.10972*, 2023.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[3] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow Transformers for high-resolution image synthesis. In *ICML*, 2024.

[4] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv:1706.02677*, 2017.

[5] Karl Heun. Neue methoden zur approximativen integration der differentialgleichungen einer unabhängigen veränderlichen. *Z. Math. Phys*, 45:23–38, 1900.

[6] 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. *NeurIPS*, 2017.

[7] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshops*, 2021.

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

[9] Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler Diffusion (SiD2): 1.5 FID on ImageNet512 with pixel-space diffusion. In *CVPR*, 2025.

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

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[12] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *NeurIPS*, 2019.

[13] 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. In *NeurIPS*, 2024.

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

[15] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. SiT: Exploring flow and diffusion-based generative models with scalable interpolant Transformers. In *ECCV*, 2024.

[16] William Peebles and Saining Xie. Scalable diffusion models with Transformers. In *ICCV*, 2023.

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

[18] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. *NeurIPS*, 29, 2016.

[19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

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

[21] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion Transformers with representation autoencoders. *arXiv:2510.11690*, 2025.