Exploring Magnitude Preservation and Rotation Modulation in Diffusion Transformers

Eric Tillmann Bill ERBILL@ETHZ.CH
ETH Zurich, Zurich Switzerland
Cristian Perez Jensen CJENSE@ETHZ.CH
ETH Zurich, Zurich Switzerland

Abstract

Denoising diffusion models exhibit remarkable generative capabilities, but remain challenging to train due to their inherent stochasticity, where high-variance gradient estimates lead to slow convergence. Previous works have shown that magnitude preservation helps with stabilizing training in the U-net architecture. This works interprets magnitude preservation as encoding the known *second moment* (unit variance) of the denoising target and ask whether this statistical prior can stabilize Diffusion Transformers (DiTs) with AdaLN conditioning. Motivated by the goal of maintaining activation magnitudes, we additionally introduce rotation modulation, which is a novel conditioning method using learned rotations instead of traditional scaling or shifting. Through empirical evaluations and ablation studies on small-scale models, we show that magnitude-preserving strategies significantly improve performance, notably reducing FID scores by $\sim$12.8%. Further, we show that rotation modulation combined with scaling is competitive with AdaLN, while requiring $\sim$5.4% fewer parameters. This work provides insights into conditioning strategies and magnitude control. Implementation available at https://github.com/ericbill21/map-dit.

Keywords: Diffusion Transformer, Magnitude Preservation, Condition Modulation.

1. Introduction

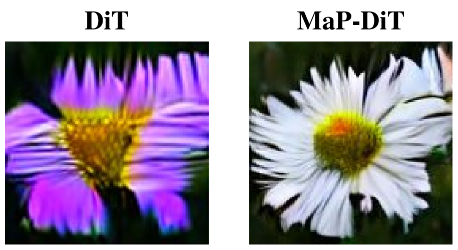


Figure 1: Effect of magnitude preservation and weight control on DiT-S/4.

Denoising diffusion models [4, 14, 15] achieve impressive generative performance but are challenging to optimize; timestep sampling, discretization, and mini-batching induce high-variance gradient estimates, slowing convergence and increasing sensitivity to schedules and architecture. At each timestep, the training target is a Gaussian noise vector with known first and second moment, *i.e.*, zero mean and unit variance. This implicitly imposes a magnitude constraint on intermediate representations. However, most diffusion backbones ignore this structure, allowing activation norms to drift and, in turn, altering the scale of gradient noise seen by the optimizer.

We hypothesize that aligning internal representations with this denoising objective yields more stable training. Concretely, preserving activation magnitudes across the network should improve the signal-to-noise ratio of stochastic gradients, and lead to more predictable optimization dynamics. Recent work by Karras et al. [5] supports this view, showing that controlling activation magnitudes layer-by-layer in the ADM architecture [1] significantly improves performance. They advocate for extending these techniques to other architectures, including transformer-based diffusion models, such as Diffusion Transformers (DiTs) [11]. This architecture replaces the conventional U-Net backbone with a transformer-based design, building on the Vision Transformer (ViT) [11].

While Karras et al. [5] focused on achieving unit activation magnitude in each ADM layer, their methods cannot be directly applied. The DiT architecture incorporates Adaptive Layer Normalization (AdaLN) modulation blocks which scale and shift inputs based on label and timestep conditioning. These modulation blocks are the only way that the conditioning variables influence the output. Thus, activation control must be extended to *arbitrary* magnitudes while preserving statistical structure.

To this end, we encode the second-moment constraint as an architectural prior. Given normalized inputs, we design a magnitude-preserving architecture and constrain weight growth so that activation magnitudes are maintained across all components, *without* any layer normalization. This links statistical structure to optimization, reducing gradient-estimator variance and widening stable step-size ranges, while keeping the parameterization simple. In addition, we introduce *rotation modulation*, an orthogonal, norm-preserving conditioning mechanism that replaces per-feature scaling/shift with learned rotations driven by timestep/label inputs. In summary, our contributions are as follows:

- We extend the magnitude preserving techniques by extending their proofs to support arbitrary magnitudes without any assumption on the underlying magnitude, enabling their application to transformer-based architectures like DiT.
- Furthermore, we conduct comprehensive ablation evaluations, showing that magnitude preservation and weight control techniques demonstrate significant performance gains across various metrics, including notable improvements in FID-10K scores.
- Lastly, we introduce a novel modulation block: *rotation modulation*. Instead of learning a scaling or translation, it learns a rotation, based on the conditioning variables. We conduct an ablation study of the three types of modulation.

2. Magnitude Preservation and Weight Growth Control

Throughout our work, we adopt the concept of *expected magnitude*, introduced by Karras et al. [5], which is defined for a random multivariate variable $\mathbf{x} \in \mathbb{R}^n$ as

$$\mathcal{M}[\mathbf{x}] \triangleq \sqrt{\frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\left[x_i^2\right]}. \tag{1}$$

The overarching goal is to ensure that every component of our modified DiT architecture operates in a magnitude-preserving manner. Specifically, for any component f, we design a magnitude preserving version f_{MP}, such that for an any input $\mathbf{x}$, we have $\mathcal{M}[f_{\text{MP}}(\mathbf{x})] = \mathcal{M}[\mathbf{x}]$. Except for the attention mechanism, we show that it is possible to generalize the magnitude-preservation techniques from Karras et al. [5] to accommodate arbitrary magnitudes $\mathcal{M}[\mathbf{x}] = \sigma$, instead of restricting them to unit magnitude $\mathcal{M}[\mathbf{x}] = 1$. Derivations for all techniques are provided in Appendix B.

Lemma 1 *Let $\mathbf{A} \in \mathbb{R}^{T\times T}$ be an unnormalized attention map and $\mathbf{V} \in \mathbb{R}^{T\times n}$ with $\mathcal{M}[\mathbf{v}_t] = \sigma$ for all $t \in [T]$. Further, define attention as*

$$\operatorname{att}(\mathbf{A}, \mathbf{V}) \triangleq \operatorname{softmax}_\beta(\mathbf{A})\mathbf{V}, \quad \operatorname{softmax}_\beta(\mathbf{A})_{ij} \triangleq \frac{\exp(a_{ij}/\beta)}{\sum_{k=1}^{n}\exp(a_{ik}/\beta)}. \tag{2}$$

Then, $\mathcal{M}[\mathrm{att}(\mathbf{A}, \mathbf{V})_t] \leq \sigma$ *for all* $t \in [T]$*. As* $\beta \to 0$*, the inequality becomes equality.*

Self-Attention. In self-attention layers, Karras et al. [5] propose to use *cosine attention* [7, 9], along with activation normalization. While cosine attention aligns with our goals of controlling activation growth, normalizing activations does not. As such, we replace the dot-product attention by cosine attention, where similarity between vectors $\mathbf{x}$ and $\mathbf{y}$ is computed by

$$\mathrm{cosim}(\mathbf{x}, \mathbf{y}) \triangleq \cos(\phi_{\mathbf{x},\mathbf{y}}) = \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} \in [-1, 1]. \tag{3}$$

Here, $\phi_{\mathbf{x},\mathbf{y}}$ denotes the angle between $\mathbf{x}$ and $\mathbf{y}$. This notion of similarity solely depends on the direction of the vectors and not on their length, preventing uncontrollable growth of activations. Lemma 1 shows that the attention mechanism only decreases the input magnitude under any attention mechanism, such as cosine and dot-product attention. Hence, we argue that normalizing activations are not necessary, since the attention mechanism cannot increase the magnitude of its input anyways.

In conclusion, under the assumptions of Lemmas 1 to 5, we are able to show for all layers that the output magnitude is equal to or upper bounded by the input magnitude. Therefore, at least at initialization when all assumptions hold true, magnitude cannot increase throughout the model.

Weight Growth Control. Karras et al. [5] showed that controlling weight growth is crucial for ensuring stable training dynamics. Because of this, we believe the same holds true for the DiT architecture. They propose using forced weight normalization, such that after each training step, each weight vector is normalized to unit magnitude. Specifically, for each linear projection with a weight matrix $\mathbf{W} \in \mathbb{R}^{n \times m}$, the normalization is applied as $\mathbf{w}_i \leftarrow \mathbf{w}_i/\|\mathbf{w}_i\|$. This operation ensures that each weight vector is normalized to lie on the unit-magnitude hypersphere. By doing so, the training gradients are effectively projected onto the tangent plane of this hypersphere, preventing the weights from growing unbounded while preserving their directionality.

3. Modulation

Following Peebles and Xie [11], we condition the latent image tokens $\mathbf{X}$ on their class labels via AdaLN. In each DiT block, AdaLN learns three vectors (scale, shift, and gate $\mathbf{s}, \mathbf{b}, \mathbf{g} \in \mathbb{R}^n$) for both the attention and MLP, and applies them as follows

$$\mathbf{x}_t \leftarrow \mathbf{g} \odot \mathrm{layer}(\mathbf{s} \odot \mathbf{x}_t + \mathbf{b}), \tag{4}$$

for each $t \in [T]$. While AdaLN is effective, it can disrupt the magnitude of the latent tokens, because the shift may move activations away from zero and the scale can amplify magnitude. Although our theoretical results remain valid under arbitrary magnitudes, a more natural and principled conditioning method that inherently preserves magnitude is desirable. To this end, we propose *rotation modulation*. Instead of scaling and shifting, it predicts a set of rotation angles and applies them to the latent tokens, thereby persevering its magnitude and respecting orthogonal symmetries.

Lemma 2 *Let* $\mathbf{R} \in \mathbb{R}^{n \times n}$ *be a rotation matrix, such that* $\mathbf{R}^\top = \mathbf{R}^{-1}$ *and* $\det(\mathbf{R}) = 1$*. Further, let* $\mathbf{x} \in \mathbb{R}^n$ *be the input vector, then*

$$\mathcal{M}[\mathbf{R}\mathbf{x}] = \mathcal{M}[\mathbf{x}]. \tag{5}$$

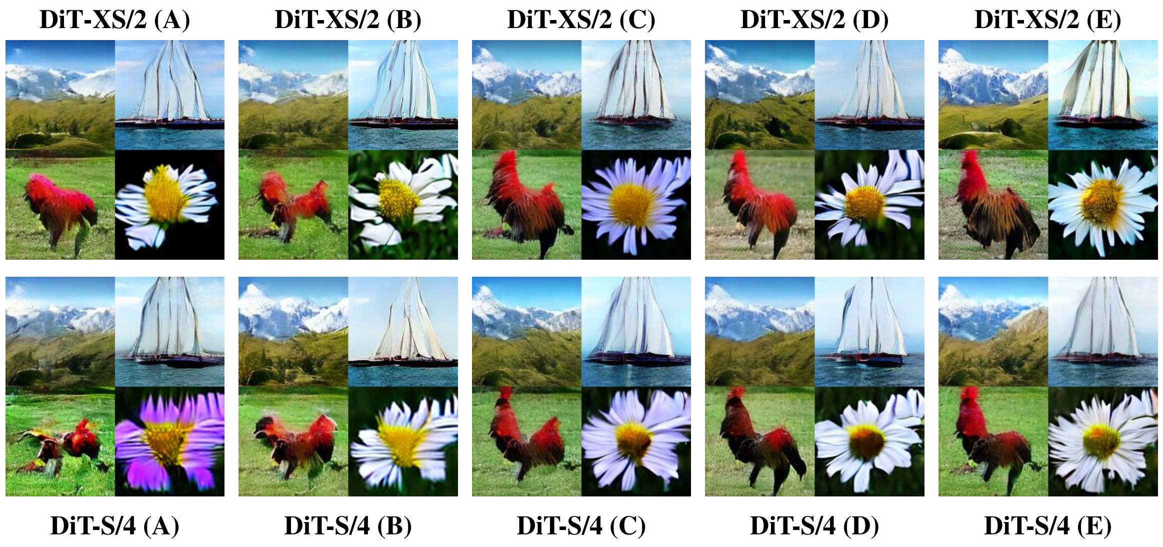


Figure 2: Four samples from all models and configurations, where all samples were generated with the same seed, a guidance scale of 5.0, and an EMA relative standard deviation of $10\,\%$. The samples include "alp" (970), "schooner" (780), "cock" (7), and "daisy" (985).

See Appendix B for a proof. Since a full n-dimensional rotation requires $^{n(n-1)}/_{2}$ degrees of freedom, we partition each token into $^{d}/_{2}$ disjoint two-dimensional sub-vectors and predict a single rotation angle per pair. Specifically, for each sub-vector $\mathbf{x}_{t,[i,i+1]} \in \mathbb{R}^2$, the model learns to predict an angle $\theta_i \in \mathbb{R}$ and modulates the sub-vector similarly to Rotary Position Embeddings (RoPE) [16]:

$$\mathbf{x}_{t,[i,i+1]} \leftarrow \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \mathbf{x}_{t,[i,i+1]}. \tag{6}$$

As a result, this "patchified" rotation requires only $^{d}/_{2}$ parameters. In contrast to previous variants, rotation modulation provides a method for conditioning on auxiliary variables, while preserving magnitude (see Lemma 2).

4. Results

In order to accommodate limited compute, we used the 128×128 variant of ImageNet [12]. To further accelerate training, we preprocessed the dataset by encoding each image with a pre-trained variational autoencoder (VAE) [6]. Further, due to limited resources, we experimented only with DiT-XS/2, DiT-S/4, and DiT-S/2 models and trained them for 400K steps. The former two were used to ablate the magnitude preservation techniques, while the latter was used for ablations of the modulation techniques.

For evaluation, we mainly consider Fréchet inception distance (FID) [3], computed over 10K samples between the ground truth dataset and model samples. In Appendix F, we also report spatial FID (sFID) [8], precision, recall, and inception scores. Note that while the use of smaller models and lower-resolution datasets limits compatibility with larger models in prior work, the focus is on analyzing architectural effects under similarly constrained model sizes.

Magnitude Preservation. We define five configurations (A–E), with Config A as the DiT baseline. Each subsequent configuration adds one of our techniques, culminating in Config E, which incorporates all modifications and removes activation normalization. For full details, consult Table 3 in Appendix C. Our results show that magnitude-preserving techniques notably improve performance. As shown in Figure 2, configurations C–E produce higher quality images compared to configurations A and B. This qualitative improvement is supported by significantly lower FID scores, as shown in Table 1. This trend holds across additional metrics such as Inception score, precision, recall, and sFID-10K. Additionally, the magnitude-preserving mechanism yields more stable gradient estimates, admitting larger *stable* learning rates in Config C–E. For a more detailed analysis, refer to Appendix F.

Table 1: FID-10K scores for all configurations.

Config	DiT-XS-2	DiT-S-4
A Baseline	99.43	103.77
B + Cosine attention	99.14	102.82
C + Magnitude preservation	**86.49**	**90.61**
D + Weight growth control	89.38	90.63
E + No normalization	86.72	91.64

However, these improvements come with increased training cost. Config E incurs an average runtime overhead of $8.5\,\%$ compared to the original DiT implementation, though when normalized by wall-clock time, Config E achieves better statistical efficiency. Additionally, we visualize the evolution of activation magnitudes before and after training in Appendix E, offering insight into the underlying causes of performance differences.

Modulation. To isolate the effect of different modulation types, we perform an ablation over all combinations of scaling, shifting, and rotation (see Section 3), using the DiT-S/2 model. Results are shown in Table 2, with full details in Table 6. Scaling provides the largest standalone benefit, significantly reducing FID. Combining scaling with either shifting or rotation improves performance further, with the best result achieved by scaling and shifting. Interestingly, using all three modulations does not yield further gains, suggesting potential interference between them. Note that rotation modulation uses half the parameters of scaling or shifting, which may explain its comparatively lower impact.

Table 2: Performance of various modulation combinations on DiT-XS/2.

Scale	Shift	Rotate	FID ↓
✓	✗	✗	72.03
✗	✓	✗	85.23
✗	✗	✓	84.62
✓	✓	✗	**69.28**
✓	✗	✓	70.86
✗	✓	✓	74.01
✓	✓	✓	72.19

5. Discussion

Our results indicate that encoding a simple statistical prior into DiT via magnitude preservation and weight-growth control reduces mini-batch gradient variance, widens the range of stable learning rates, and improves notable metrics, including FID. Rotation modulation further shows that symmetry-respecting conditioning can stabilize training while remaining parameter-efficient. A current limitation is the magnitude-preserving SiLU activation, which cannot be naturally extended to arbitrary magnitudes. Future work could experiment with other activation functions, such as Leaky ReLU. For this, we provide a magnitude-preserving version in Lemma 6. Additionally, future work could investigate scaling these results up to larger models, as well as text-to-image models.

References

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

[2] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

[3] 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.

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

[5] 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.

[6] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[7] Chunjie Luo, Jianfeng Zhan, Xiaohe Xue, Lei Wang, Rui Ren, and Qiang Yang. Cosine normalization: Using cosine similarity instead of dot product in neural networks. In *International conference on artificial neural networks and machine learning*, pages 382–391. Springer, 2018.

[8] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter Battaglia. Generating images with sparse representations. In *International conference on machine learning*, pages 7958–7968, 2021.

[9] Quang-Huy Nguyen, Cuong Q Nguyen, Dung D Le, and Hieu H Pham. Enhancing few-shot image classification with cosine transformer. *IEEE Access*, 11:79659–79672, 2023.

[10] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[11] 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.

[12] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International journal of computer vision*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[13] 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.

[14] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[15] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International conference on learning representations*, 2021.

[16] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Appendix Contents

Appendix A. Implementation Details

In line with Karras et al. [5], we remove all biases from linear layers. To restore the capability of the model learning a bias, we concatenate ones to the channels of the model's input. Furthermore, we implement cosine attention by normalizing the queries and keys before computing their inner products. This is equivalent and computationally more efficient.

Appendix B. Proofs

In this section, we present the remaining lemmas and proofs, demonstrating that each model component preserves or upper bounds the magnitude of its input. Our results generalize Karras et al. [5] to arbitrary input magnitudes.

B.1. Linear Layer

Lemma 3 *A linear layer preserves the magnitude of its input by normalizing the rows to euclidean unit. In particular, let $\mathbf{W} \in \mathbb{R}^{m \times n}$ and $\mathbf{x} \in \mathbb{R}^n$ an input vector. Assume that the input features are* i.i.d.*, such that $\mathbb{E}[x_i x_j] = 0$ for all $i \neq j$ and $\mathbb{E}[x_i^2] = \sigma^2$ for all i. Denote by $\hat{\mathbf{W}}$ the row-wise normalized version of $\mathbf{W}$, defined as $\hat{w}_{ij} = w_{ij}/\|\mathbf{w}_i\|_2$, where $\mathbf{w}_i$ is the i-th row of $\mathbf{W}$. Then*

$$\mathcal{M}\left[\hat{\mathbf{W}}\mathbf{x}\right] = \sigma. \tag{7}$$

Proof Define $\hat{\mathbf{W}}$ and $\mathbf{x}$ as in Lemma 3, then

$$\begin{aligned}
\mathcal{M}\left[\hat{\mathbf{W}}\mathbf{x}\right]^2 &= \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[\left(\sum_{j=1}^{n}\hat{w}_{ij}x_j\right)^2\right] &(8)\\
&= \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n}\hat{w}_{ij}^2\mathbb{E}\left[x_j^2\right] &(9)\\
&= \frac{1}{m}\sum_{i=1}^{m}\frac{\sigma^2}{\|\mathbf{w}_i\|_2^2}\sum_{j=1}^{n}w_{ij}^2 &(10)\\
&= \frac{1}{m}\sum_{i=1}^{m}\sigma^2 &(11)\\
&= \sigma^2. &(12)
\end{aligned}$$

Thus, the normalized projection preserves the magnitude of the input. ■

In practice, the normalization of $\mathbf{W}$ is performed in every forward pass to ensure that the rows have unit norm, maintaining the desired property of magnitude preservation. Furthermore, Lemma 3 extends to linear embeddings, which are equivalent to linear projections when inputs are one-hot encoded.

B.2. Residual Connection

Lemma 4 *A residual connection preserves the magnitude of two inputs for any magnitude by scaling its output. In particular, assume $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ are uncorrelated,* i.e., $\mathbb{E}[x_i y_j] = 0, \forall i, j \in [n]$. *Further, assume $\mathcal{M}[\mathbf{x}] = \sigma$ and $\mathcal{M}[\mathbf{y}] = \tau$. Let $\alpha \in [0, 1]$, then*

$$\mathcal{M}\big[\sqrt{\alpha}\mathbf{x} + \sqrt{1-\alpha}\mathbf{y}\big]^2 = \alpha\sigma^2 + (1-\alpha)\tau^2. \tag{13}$$

As a special case, if $\mathcal{M}[\mathbf{x}] = \mathcal{M}[\mathbf{y}] = \sigma$, then

$$\mathcal{M}\big[\sqrt{\alpha}\mathbf{x} + \sqrt{1-\alpha}\mathbf{y}\big] = \sigma. \tag{14}$$

Proof Define $\mathbf{x}$, $\mathbf{y}$, and α as in Lemma 4. Then,

$$\begin{aligned}
&\mathcal{M}\big[\sqrt{\alpha}\mathbf{x} + \sqrt{1-\alpha}\mathbf{y}\big]^2 && (15)\\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\Big[\big(\sqrt{\alpha}x_i + \sqrt{1-\alpha}y_i\big)^2\Big] && (16)\\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\big[\alpha x_i^2 + (1-\alpha)y_i^2 + \sqrt{\alpha}\sqrt{1-\alpha}x_i y_i\big] && (17)\\
&= \frac{\alpha}{n}\sum_{i=1}^{n}\mathbb{E}[x_i^2] + \frac{1-\alpha}{d}\sum_{i=1}^{n}\mathbb{E}[y_i^2] && (18)\\
&= \alpha\mathcal{M}[\mathbf{x}]^2 + (1-\alpha)\mathcal{M}[\mathbf{y}]^2 && (19)\\
&= \alpha\sigma^2 + (1-\alpha)\tau^2. && (20)
\end{aligned}$$

The special case follows as an application of the above. ■

B.3. Sigmoid Linear Unit (SiLU)

Lemma 5 *The sigmoid linear unit (SiLU) [2] activation function can preserve its input magnitude by a scaling. In particular, define $s : \mathbb{R} \to \mathbb{R}$ as*

$$s(\sigma) \triangleq \frac{\sigma}{\sqrt{\mathbb{E}_{z\sim\mathcal{N}(0,\sigma)}[\text{silu}^2(z)]}}. \tag{21}$$

Let $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_n)$ (note $\mathcal{M}[\mathbf{x}] = \sigma$) then the SiLU activation function scaled by $s(\sigma)$ preserves the magnitude of its input,

$$\mathcal{M}[s(\sigma)\cdot\text{silu}(\mathbf{x})] = \sigma. \tag{22}$$

Proof Define $\mathbf{x}$, $s : \mathbb{R} \to \mathbb{R}$, and silu $: \mathbb{R} \to \mathbb{R}$ as in Lemma 5. Then,

$$\begin{aligned}
\mathcal{M}[s(\sigma)\cdot\text{silu}(x)]^2 &= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\Big[\big(s(\sigma)\cdot\text{silu}(x_i)\big)^2\Big] && (23)\\
&= s^2(\sigma)\cdot\mathbb{E}_{z\sim\mathcal{N}(0,\sigma)}\big[\text{silu}(z)^2\big] && (24)\\
&= \sigma^2. && (25)
\end{aligned}$$

Hence, the scaled activation function preserves its input's magnitude. ■

In order to ease computation, we make the assumption that the input is sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I}_n)$. In this case, the scaling factor $s(1)$ can be computed numerically and yields

$$s(1) \approx \frac{1}{0.596}. \tag{26}$$

Thus, the magnitude-preserving SiLU activation function is evaluated as ${}^{\text{silu}(\cdot)}/_{0.596}$.

B.4. Leaky Rectified Linear Unit (ReLU)

Lemma 6 *The Leaky ReLU activation function,*

$$\text{lrelu}_\alpha(x) \triangleq \begin{cases} x & x \geq 0 \\ \alpha x & x < 0, \end{cases} \tag{27}$$

can preserve the input magnitude through appropriate scaling. Specifically, let $\alpha \in \mathbb{R}^+$ be the negative slope parameter, and let $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$. Then,

$$\mathcal{M}\left[\left(\sqrt{\frac{2}{\alpha^2+1}}\right) \text{lrelu}_\alpha(\mathbf{x})\right] = \mathcal{M}[\mathbf{x}]. \tag{28}$$

Proof Define $\mathbf{x}$ and α as in Lemma 6. Then,

$$\begin{aligned}
\mathcal{M}[\text{lrelu}(\mathbf{x})]^2 &= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{x_i \sim \mathcal{N}(0,\sigma^2)}\left[\text{lrelu}_\alpha(x_i)^2\right] && (29)\\
&= \mathbb{E}_{x \sim \mathcal{N}(0,\sigma^2)}\left[\text{lrelu}_\alpha(x)^2\right] && (30)\\
&= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{1}{2\sigma^2}x^2\right) \cdot \text{lrelu}_\alpha(x)^2 dx && (31)\\
&= \frac{1}{\sqrt{2\pi\sigma^2}}\left(\int_{-\infty}^{0} \exp\left(-\frac{1}{2\sigma^2}x^2\right)\alpha^2 x^2 dx + \int_{0}^{\infty} \exp\left(-\frac{1}{2\sigma^2}x^2\right)x^2 dx\right) && (32)\\
&= \frac{1}{\sqrt{2\pi\sigma^2}}\left(\alpha^2\int_{0}^{\infty} \exp\left(-\frac{1}{2\sigma^2}x^2\right)x^2 dx + \int_{0}^{\infty} \exp\left(-\frac{1}{2\sigma^2}x^2\right)x^2 dx\right) && (33)\\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \cdot (\alpha^2+1)\int_{0}^{\infty} \exp\left(-\frac{1}{2\sigma^2}x^2\right) \cdot x^2 dx && (34)\\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \cdot (\alpha^2+1) \cdot \frac{\sqrt{\pi}}{4 \cdot \left(\frac{1}{2\sigma^2}\right)^{3/2}} && (35)\\
&= \left(\frac{\alpha^2+1}{2}\right)\sigma^2, && (36)
\end{aligned}$$

where in Equation (34) we use the fact that the integral is a scaled gamma function. ■

As a special case, setting $\alpha = 0$ yields the standard ReLU function. In this case, the following corollary holds:

Corollary 7 *The ReLU activation function preserves the input magnitude under appropriate scaling. Specifically, if $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$, then*

$$\mathcal{M}\left[\sqrt{2} \cdot \mathrm{relu}(\mathbf{x})\right] = \mathcal{M}[\mathbf{x}].$$

B.5. Attention

Proof [Proof of Lemma 1] Define $\mathbf{A} \in \mathbb{R}^{T \times T}$ and $\mathbf{V} \in \mathbb{R}^{T \times n}$ as in Lemma 1. We need to show that the attention mechanism preserves the magnitude of the values. Attention first normalizes the attention map,

$$\mathbf{B} = \mathrm{softmax}_\beta(\mathbf{A}), \tag{37}$$

where softmax is defined as in Lemma 1. Now, the rows of $\mathbf{B}$ are normalized to be distributions over values, *i.e.*, $\mathbf{b}_i \in \Delta^{T-1}$, where Δ^{m-1} is the m-dimensional probability simplex.

Here, the rows are normalized to proper distributions over values. The output is computed by

$$\mathrm{att}(\mathbf{A}, \mathbf{V}) = \mathbf{B}\mathbf{V}, \quad \mathrm{att}(\mathbf{A}, \mathbf{V})_i = \sum_{t=1}^{T} b_{it} \mathbf{v}_t. \tag{38}$$

We can now show that the output magnitude is upper bounded by the input magnitude σ,

$$\begin{aligned}
\mathcal{M}[\mathrm{att}(\mathbf{A}, \mathbf{V})_i]^2 &= \frac{1}{n} \sum_{j=1}^{d} \mathbb{E}\left[\mathrm{att}(\mathbf{A}, \mathbf{V})_{ij}^2\right] && (39)\\
&= \frac{1}{n} \sum_{j=1}^{d} \mathbb{E}\left[\left(\sum_{t=1}^{T} b_{it} v_{tj}\right)^2\right] && (40)\\
&\leq \frac{1}{n} \sum_{j=1}^{n} \mathbb{E}\left[\sum_{t=1}^{T} b_{it} v_{tj}^2\right] && (41)\\
&= \sum_{t=1}^{T} b_{it} \cdot \frac{1}{n} \sum_{j=1}^{n} \mathbb{E}\left[v_{tj}^2\right] && (42)\\
&= \sum_{t=1}^{T} b_{it} \sigma^2 && (43)\\
&= \sigma^2, && (44)
\end{aligned}$$

where Jensen's inequality is applied in Equation (41), because $f(x) = x^2$ is a convex function and $\mathbf{b}_i \in \Delta^{T-1}$. If $b_{it} = 1$ for some $t \in [T]$ (and hence the other timesteps have weight 0), it is trivial to see that the inequality becomes equality. ■

B.6. Rotation Modulation

Proof [Proof of Lemma 2] Define $\mathbf{R}$ and $\mathbf{x}$ as in Lemma 2. Then,

$$\begin{aligned}
\mathcal{M}[\mathbf{R}\mathbf{x}]^2 &= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\Big[(\mathbf{R}\mathbf{x})_i^2\Big] && (45)\\
&= \frac{1}{n}\mathbb{E}\left[\sum_{i=1}^{n}(\mathbf{R}\mathbf{x})_i^2\right] && (46)\\
&= \frac{1}{n}\mathbb{E}\left[\|\mathbf{R}\mathbf{x}\|_2^2\right] && (47)\\
&= \frac{1}{n}\mathbb{E}\Big[\mathbf{x}^\top\mathbf{R}^\top\mathbf{R}\mathbf{x}\Big] && (48)\\
&= \frac{1}{n}\mathbb{E}\Big[\mathbf{x}^\top\mathbf{x}\Big] && (49)\\
&= \frac{1}{n}\mathbb{E}\left[\sum_{i=1}^{n}x_i^2\right] && (50)\\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[x_i^2\right] && (51)\\
&= \mathcal{M}[\mathbf{x}]. && (52)
\end{aligned}$$

This concludes the proof. ■

Appendix C. Configurations and Reproducibility

A complete list of training hyperparameters is included in Table 5, and detailed configurations for each setup in the magnitude preservation ablation study are summarized in Table 3. Estimated training run-times for all evaluated models are reported in Table 4. For the conditioning ablation study, all corresponding hyperparameters and results are documented in Table 6.

Table 3: Configuration Attributes. A check (✓) indicates that a particular training attribute is enabled.

Attribute	A	B	C	D	E
Cosine attention	✗	✓	✓	✓	✓
Weight norm	✗	✗	✓	✓	✓
MP embedding	✗	✗	✓	✓	✓
MP pos enc	✗	✗	✓	✓	✓
MP residual	✗	✗	✓	✓	✓
MP SiLU	✗	✗	✓	✓	✓
Forced weight norm	✗	✗	✗	✓	✓
No layer norm	✗	✗	✗	✗	✓

Table 4: Runtime Overview, where each model ran with batch size 256 under Config E on a GTX 1080 Ti.

Model	Params (M)	Steps/sec	Time (h)
DiT-B/4	131.4	2.13	52.2
DiT-B/8	131.9	4.11	27.0
DiT-S/2	33.2	2.11	52.7
DiT-S/4	33.3	5.65	19.7
DiT-S/8	33.5	7.91	14.0
DiT-XS/2	7.9	6.74	16.5
DiT-XS/4	7.9	11.6	9.6
DiT-XS/8	8.0	14.9	7.5

Table 5: Hyperparameters for ablation study. Each row summarizes training settings for DiT-XS/2 (top) and DiT-S/4 (bottom) across configurations A to E.

	Config	#EMA Snapshots	EMA Stdevs	Warm-up Steps	LR Decay Start Step	#Params (M)	Train Steps (K)	Batch Size	Learning Rate
DiT-XS/2	Config A	250	$\{0.05, 0.1\}$	2666	40000	7.60	400	256	1×10^{-4}
	Config B	250	$\{0.05, 0.1\}$	2666	40000	7.60	400	256	1×10^{-4}
	Config C	250	$\{0.05, 0.1\}$	2666	40000	7.86	400	256	1×10^{-2}
	Config D	250	$\{0.05, 0.1\}$	2666	40000	7.86	400	256	1×10^{-2}
	Config E	250	$\{0.05, 0.1\}$	2666	40000	7.86	400	256	1×10^{-2}
DiT-S/4	Config A	250	$\{0.05, 0.1\}$	2666	40000	32.89	400	256	1×10^{-4}
	Config B	250	$\{0.05, 0.1\}$	2666	40000	32.89	400	256	1×10^{-4}
	Config C	250	$\{0.05, 0.1\}$	2666	40000	33.28	400	256	1×10^{-2}
	Config D	250	$\{0.05, 0.1\}$	2666	40000	33.28	400	256	1×10^{-2}
	Config E	250	$\{0.05, 0.1\}$	2666	40000	33.28	400	256	1×10^{-2}

Table 6: Hyperparameters and results for conditioning ablation study. Each row summarizes the training settings for DiT-S/2. Across all runs, we used 250 EMA snapshots with standard deviations {0.05, 0.1}, 2666 warm-up steps, learning rate decay starting at 40K steps, and a total of 400K training steps. The batch size was set to 256, and the learning rate was fixed at 1×10^{-4}.

	Scale	**Shift**	**Rotate**	**Params (M)**	**Steps/s**	**FID** $\downarrow$	**sFID** $\downarrow$	**IS** $\downarrow$	**Precision** $\uparrow$	**Recall** $\uparrow$
DiT-S/2	✓	✗	✗	29.1	2.43	72.03	13.38	43.77	0.185	0.394
	✗	✓	✗	25.6	2.62	85.23	73.87	11.12	0.145	0.387
	✗	✗	✓	25.6	2.22	84.62	73.45	11.33	0.147	0.379
	✓	✓	✗	32.8	2.34	69.28	47.17	14.21	0.186	0.426
	✓	✗	✓	31.0	2.12	70.86	50.78	13.67	0.191	0.405
	✗	✓	✓	27.4	2.21	74.01	57.60	12.82	0.177	0.397
	✓	✓	✓	34.6	2.00	72.19	51.87	14.23	0.180	0.423

Appendix D. Power-Function Based EMA

Exponential moving average (EMA) has proven to be highly effective in image generation [15], but their performance is very sensitive to the decay parameter [10]. Let $\boldsymbol{\theta}(t)$ denote the model parameters at step t, and $\hat{\boldsymbol{\theta}}_\beta(t)$ their EMA. The traditional EMA fixes a decaying parameter $\beta \in [0, 1]$ and updates as

$$\hat{\boldsymbol{\theta}}_\beta(t) = \beta\hat{\boldsymbol{\theta}}_\beta(t-1) + (1-\beta)\boldsymbol{\theta}(t). \tag{53}$$

However, this places non-negligible weight on the random initialization of the first few training steps. Ideally, the decay rate should be small initially to reduce noise and grow larger to smooth the final parameters.

To address the aforementioned issues, Karras et al. [5] proposes using a power-function based EMA. We present our own derivation, closely aligned with its practical implementation and simpler to interpret,

$$\hat{\boldsymbol{\theta}}_\gamma(t) = \frac{1}{Z(t)} \sum_{\tau=0}^{t} \tau^\gamma \boldsymbol{\theta}(\tau), \quad Z(t) = \sum_{\tau=0}^{t} \tau^\gamma \tag{54}$$

where $Z(t)$ ensures the weights sum to 1, and γ is a hyperparameter that controls the overall decay, so that large γ places more weight on recent steps. To make γ more intuitive, Karras et al. [5] parametrize it via its relative standard deviation $\sigma_{\text{rel}} = (\gamma+1)^{\frac{1}{2}}(\gamma+2)^{-1}(\gamma+3)^{-\frac{1}{2}}$, which gauges the "width" of the weighting distribution relative to the training time. Visualizations of different decay parameters are presented in Figure 3.

In the following, we will prove closed-form update formula for the power function based EMA. For this, let γ be fixed, and $\hat{\boldsymbol{\theta}}_\gamma(t)$ to denote the EMA and $\boldsymbol{\theta}(t)$ the parameters at step t. Then the definition of the EMA is equivalent to the following,

$$\hat{\boldsymbol{\theta}}_\gamma(t) \triangleq \frac{1}{Z(t)} \sum_{\tau=0}^{t} \tau^\gamma \boldsymbol{\theta}(\tau) \tag{55}$$

$$= \frac{Z(t-1)}{Z(t)} \frac{1}{Z(t-1)} \sum_{\tau=0}^{t-1} \tau^\gamma \boldsymbol{\theta}(\tau) + \frac{t^\gamma}{Z(t)} \boldsymbol{\theta}(t) \tag{56}$$

$$= \frac{Z(t-1)}{Z(t)} \hat{\boldsymbol{\theta}}_\gamma(t-1) + \frac{Z(t)-Z(t-1)}{Z(t)} \boldsymbol{\theta}(t) \tag{57}$$

$$= \frac{Z(t-1)}{Z(t)} \hat{\boldsymbol{\theta}}_\gamma(t-1) + \left(1 - \frac{Z(t-1)}{Z(t)}\right) \boldsymbol{\theta}(t) \tag{58}$$

Unlike standard EMA, the decay factor $Z(t-1)/Z(t)$ now depends on t, making it small early in training and close to to 1 for large t.

However, calculating the decaying term directly can be numerically unstable, as the complexity scales linearly with the step t, as its just a sum over powers

$$Z(t) = \sum_{\tau=0}^{t} \tau^\gamma. \tag{59}$$

To mitigate this, we approximate each sum via a continuous integral, which consequently can be represented in closed form,

$$\frac{Z(t-1)}{Z(t)} \approx \frac{\int_0^{t-1} \tau^\gamma \mathrm{d}\tau}{\int_0^t \tau^\gamma \mathrm{d}\tau} = \frac{\frac{1}{\gamma+1}(t-1)^{\gamma+1}}{\frac{1}{\gamma+1} t^{\gamma+1}} = \left(1 - \frac{1}{t}\right)^{\gamma+1}. \tag{60}$$

This derivation follows the same ideas as Karras et al. [5], but is presented here in a form closer to our actual implementations, and allows for easier understanding. The resulting update formula matches the one in Karras et al. [5] exactly.

Terminology While this approach shares similarities with traditional EMA, it does not follow a strictly "exponential" decay due to its dependence on the power function. Therefore, referring to it as an EMA can be misleading. However, for consistency with prior work and to emphasize its utility in improving training dynamics, we retain the term EMA to avoid unnecessary confusion.

Post hoc EMA The choice of decay parameter strongly affects performance [10], and the same holds for our hyperparameter γ. Ideally, one could choose γ (or equivalently σ_{rel}) *after* training. Karras et al. [5] show that it is possible to reconstruct different EMA profiles accurately from a single training run, by storing a small set of model snapshots under varying γ. Specifically, we store two EMA snapshots for $\gamma_1 = 16.97$ and $\gamma_2 = 6.94$ (corresponding to $\sigma_{\text{rel}} = 0.05$ and 0.1, respectively) every 1600 training steps. After processing 400k images. To save storage, we store each snapshot in 16-bit floating point precision.

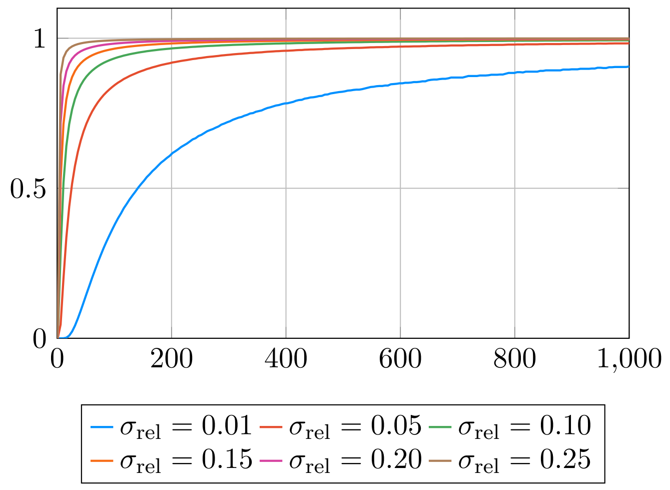


Figure 3: EMA Decaying Factor Curves. Decaying factor $^{Z(t-1)}/_{Z(t)}$ (y-axis) over the first 1000 steps (x-axis) for various relative standard deviations.

For an arbitrary γ^* value, we can approximate $\hat{\boldsymbol{\theta}}_{\gamma^*}$ after training by computing the least-squares fit between the two stored EMA profiles and the desired γ^*, then combine the snapshots accordingly.

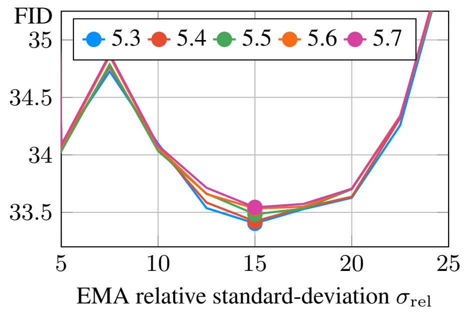


Figure 4: FID-10K across varying σ_{rel} values. Experiments were conducted using DiT-XS/2 using Config E.

Results The ability to reconstruct different EMA profiles post-training adds flexibility to this approach. In Figure 4, we calculated FID-10K scores for DiT-XS/2 in Config E using various reconstructed EMA profiles. The optimal FID-10K score was achieved with $\sigma_{\text{rel}} = 15\,\%$, improving the score by ≈ 0.64 for all tested guidance scales. Interestingly, the optimal σ_{rel} was not saved during training.

Appendix E. Activation Magnitude Evolution

To assess the impact of our magnitude-preserving techniques, we visualize the evolution of activation magnitudes across the DiT blocks of DiT-S/4 in Figure 5.

At initialization, Config A maintains a constant magnitude across blocks. However, after 400k training steps, it exhibits a clear trend of increasing magnitudes at the outputs of both the self-attention and MLP modules. Despite this accumulation, LayerNorm ensures that the inputs to these modules remain consistently near zero magnitude, stabilizing the model.

In contrast, Config E, which applies all proposed magnitude-preserving techniques and omits LayerNorm, shows a steady decline in magnitude at initialization. This behavior aligns with our theoretical result in Lemma 1, which shows that attention layers inherently reduce magnitude. After training, Config E decrease magnitudes steadily in all blocks and avoids the excessive growth as seen in Config A, suggesting that our techniques are effective even without normalization.

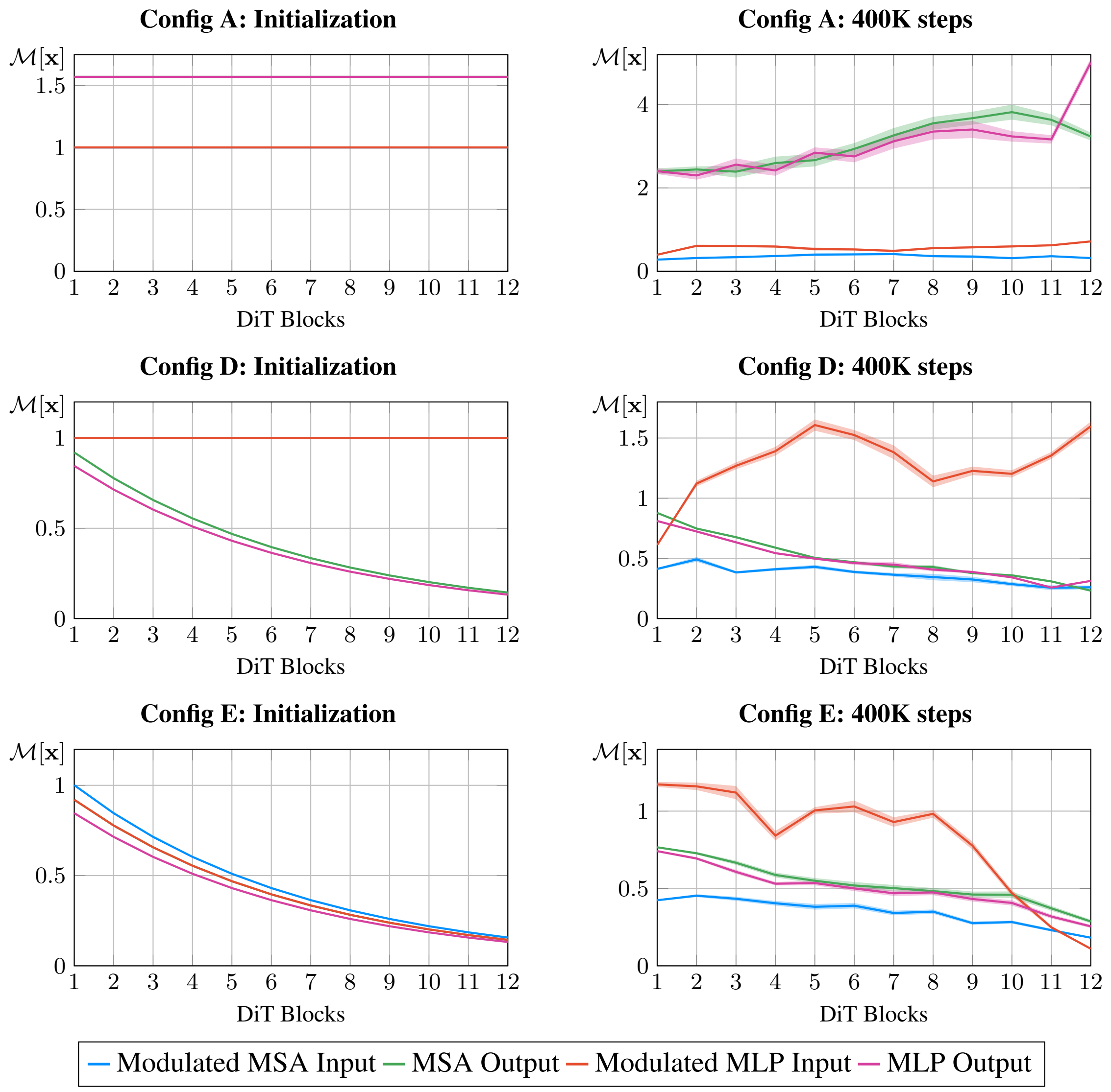


Figure 5: Activation magnitude evolution across DiT blocks in DiT-S/4. For blocks 1–12, we show mean activation magnitudes (averaged over all labels and timesteps), with shaded areas representing ± 3 standard deviations. Plotted are the AdaLN-modulated input (see Section 3) and the output after the residual connection, for both the self-attention and MLP modules. Results are shown at initialization (left) and after 400K training steps (right) for each configuration.

We also include Config D, which uses the same techniques as Config E but retains LayerNorm. At initialization, the only difference is that inputs are normalized to unit magnitude by LayerNorm. After training, its magnitude evolution closely resembles that of Config E, with the main distinction being a spike in the MLP input due to LayerNorm. Given the similarity in both behavior and performance (see Section 4), we conclude that LayerNorm may not be necessary when designing DiT in a magnitude-preserving manner.

Appendix F. Convergence

To demonstrate the training process and verify that the choice of 400K training steps was not arbitrary, we evaluated several metrics for Configurations A to E on both DiT-XS/2 and DiT-S/4. Specifically, we used the training checkpoints saved without EMA at intervals of every 50k steps. For each model and configuration combination, the metrics were calculated both in unguided mode and with guidance using a guidance scale of 5.0. The metrics include

Inception Score	Measures the quality and diversity of generated images based on the output probabilities of a pretrained classifier. Higher scores indicate better generative performance [13].
Precision/Recall	Quantify the alignment between generated and real data distribution. Precision measures how much of the generated data falls within the real data manifold, while recall assesses how much of the real data is covered by the generated data.
FID-10K	The Fréchet inception distance (FID) [3] computes the similarity between the real and generated data distributions using the mean and covariance of their feature representations. Lower values indicate higher similarity. FID-10K calculates this score over 10,000 samples [3].
sFID-10K	sFID is a variant of FID that reduces computational complexity by using a single set of real features for comparison. Like FID, lower values indicate better alignment between real and generated data distributions [8].

Plots showing the progression of these metrics over the training span are provided. For DiT-XS/2, unguided and guided results can be found in Figure 6. Similarly, the results for DiT-S/4 are presented in Figure 7.

Appendix G. Samples

Figures 8 to 14 show samples from all trained models under the same seed. This make it possible to directly compare the generations of the different models. They were all generated with guidance scale 5.0 and EMA σ_{rel} of 10%. They are best viewed zoomed in.

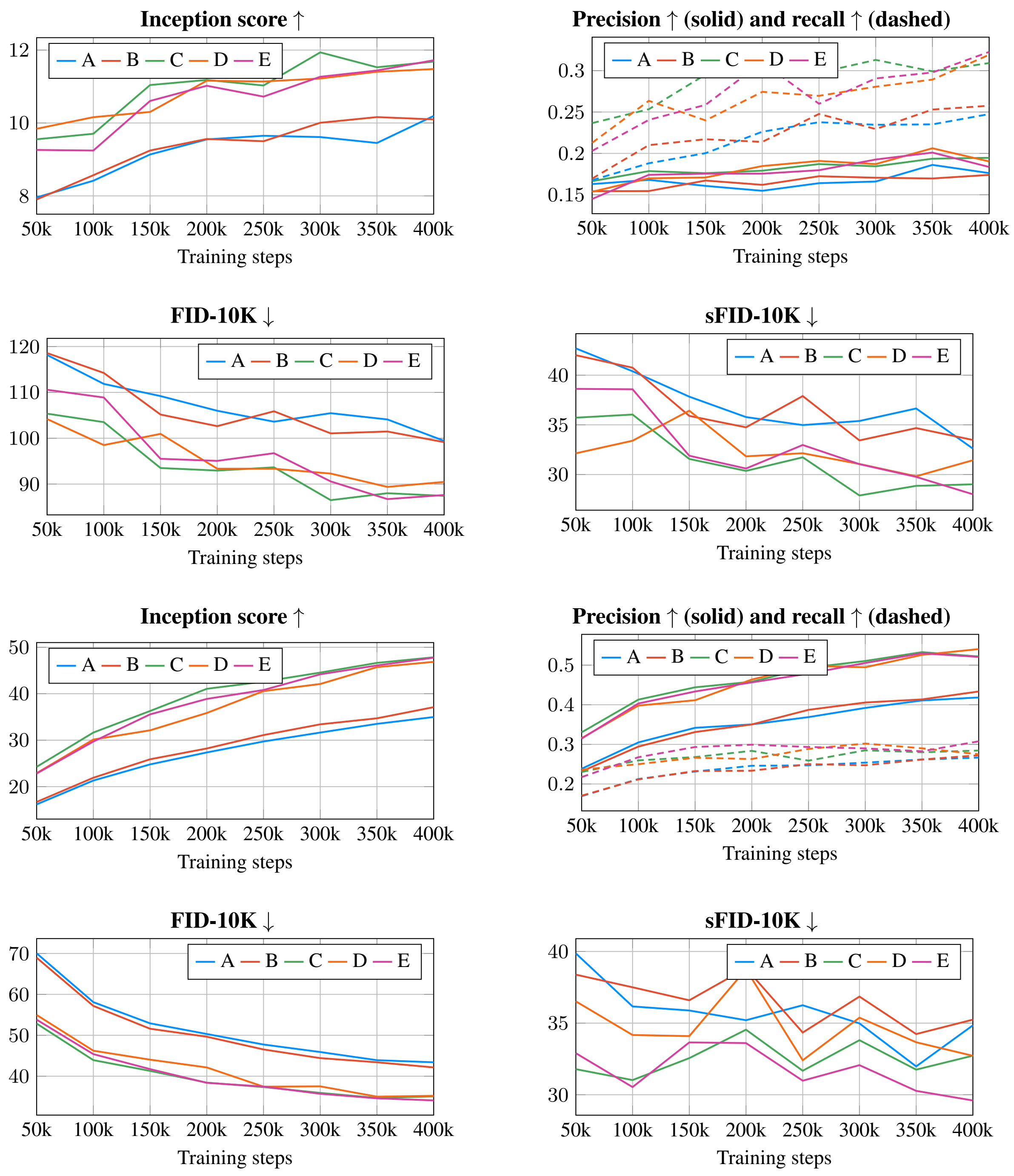


Figure 6: **Convergence trends of generative metrics for DiT-XS/2 without guidance (top) and with guidance scale** 5.0 **(bottom).** Metrics including the Inception Score, Precision, Recall, FID-10k, and sFID-10k are calculated for all configurations using a guidance scale of 5.0. These evaluations are performed on model checkpoints taken every 50k training steps.

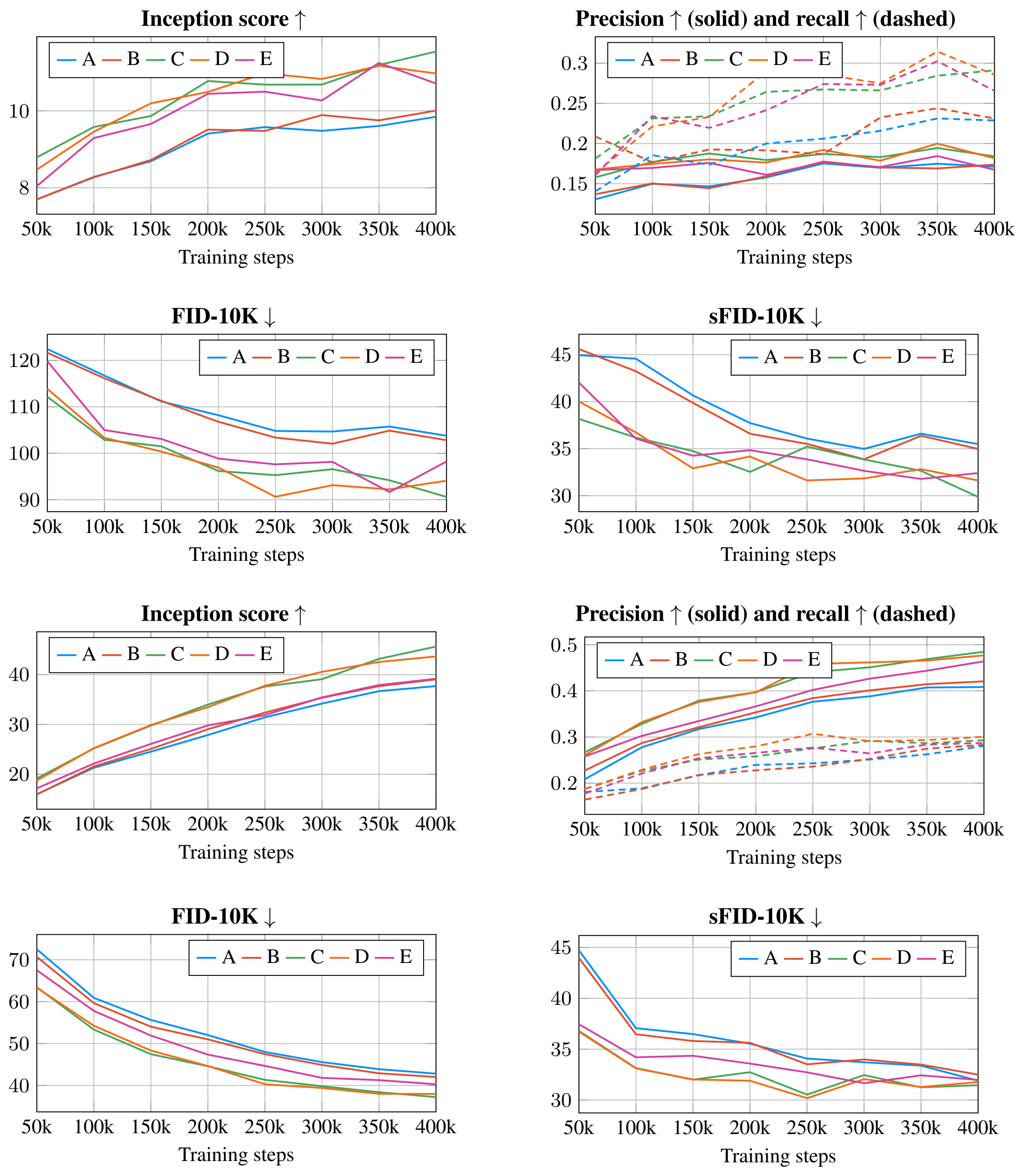
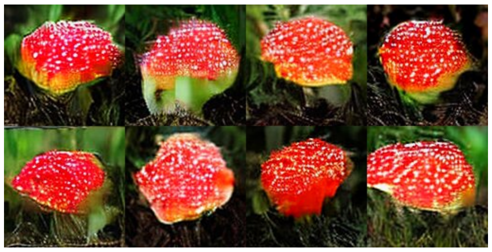
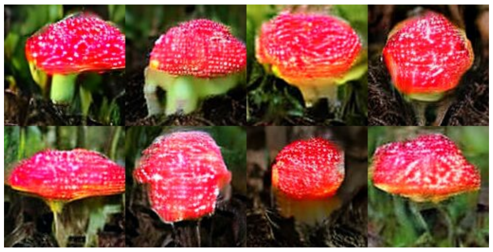
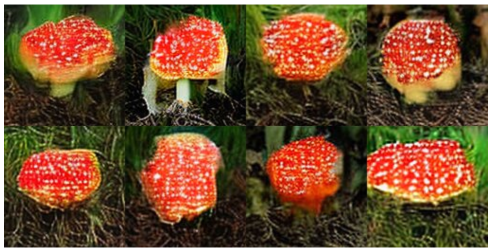
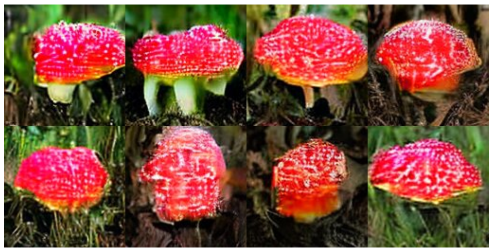


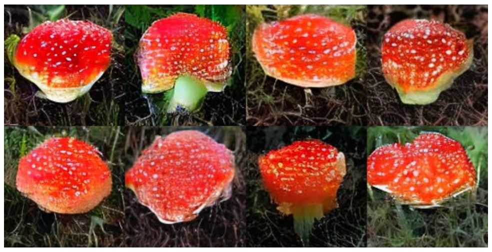
Figure 7: Convergence trends of generative metrics for DiT-S/4 without guidance (top) and with guidance scale 5.0 (bottom). Metrics including the Inception Score, Precision, Recall, FID-10k, and sFID-10k are calculated for all configurations using a guidance scale of 5.0. These evaluations are performed on model checkpoints taken every 50k training steps.

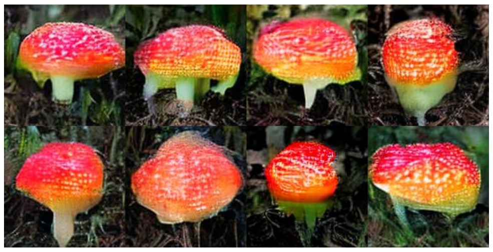

DiT-XS/2 (Config A).

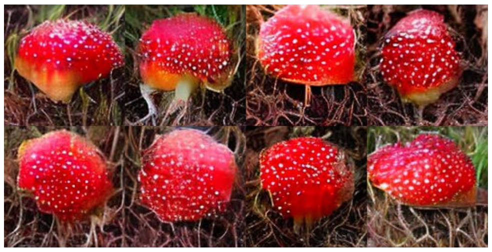

DiT-S/4 (Config A).

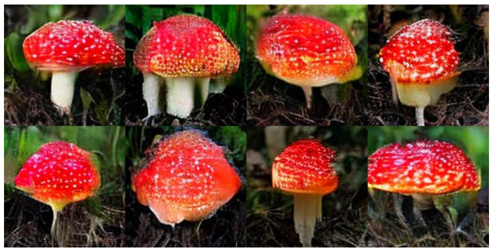

DiT-XS/2 (Config B).

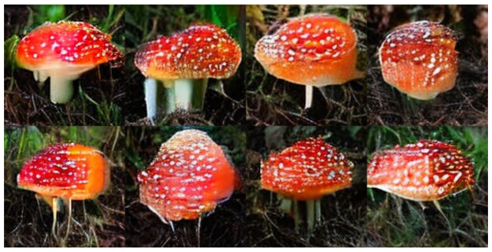

DiT-S/4 (Config B).


DiT-XS/2 (Config C).


DiT-S/4 (Config C).


DiT-XS/2 (Config D).


DiT-S/4 (Config D).


DiT-XS/2 (Config E).

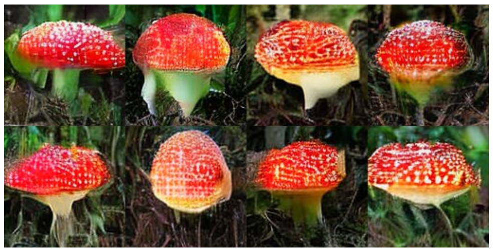
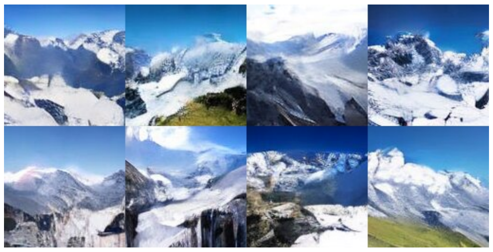

DiT-S/4 (Config E).

Figure 8: Uncurated samples of DiT-XS/2 and DiT-S/4 across all configurations.
Guidance scale = 5.0
EMA relative standard-deviation = $10\,\%$
Class label = “Agaric” (992)

DiT-XS/2 (Config A).


DiT-S/4 (Config A).

DiT-XS/2 (Config B).

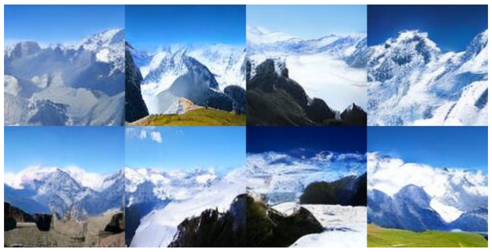
DiT-S/4 (Config B).

DiT-XS/2 (Config C).

DiT-S/4 (Config C).

DiT-XS/2 (Config D).

DiT-S/4 (Config D).


DiT-XS/2 (Config E).

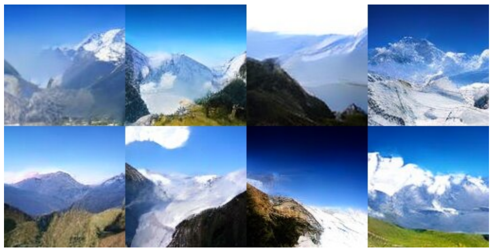

DiT-S/4 (Config E).

Figure 9: Uncurated samples of DiT-XS/2 and DiT-S/4 across all configurations.
Guidance scale = 5.0
EMA relative standard-deviation = 10 %
Class label = "Alp" (970)

DiT-XS/2 (Config A).

DiT-S/4 (Config A).

DiT-XS/2 (Config B).

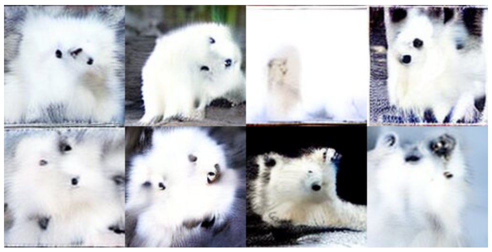
DiT-S/4 (Config B).

DiT-XS/2 (Config C).

DiT-S/4 (Config C).

DiT-XS/2 (Config D).

DiT-S/4 (Config D).


DiT-XS/2 (Config E).

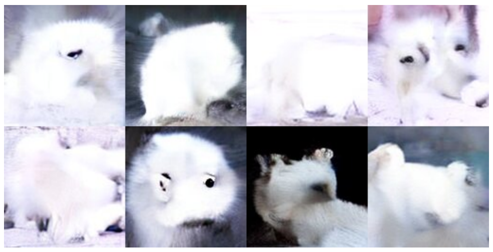
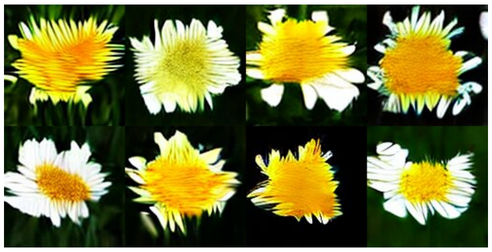
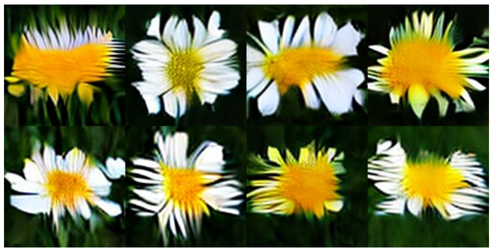
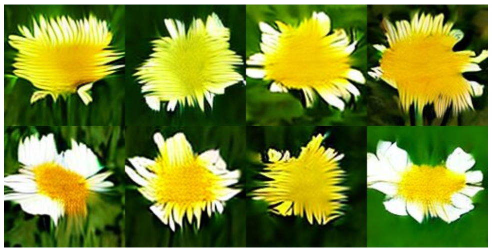
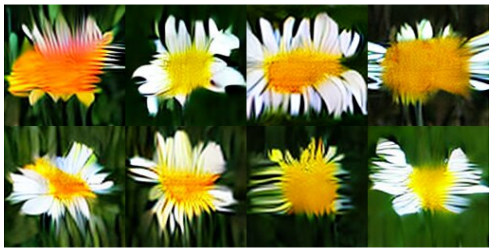

DiT-S/4 (Config E).

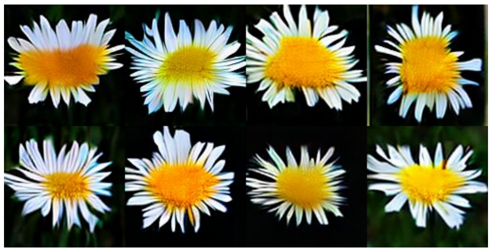
Figure 10: Uncurated samples of DiT-XS/2 and DiT-S/4 across all configurations.
Guidance scale = 5.0
EMA relative standard-deviation = $10\,\%$
Class label = “Arctic fox” (279)

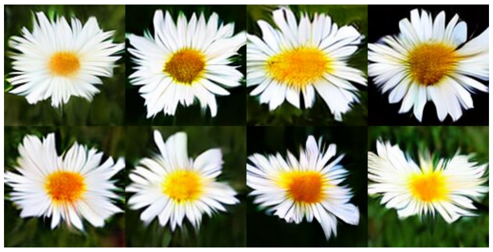

DiT-XS/2 (Config A).

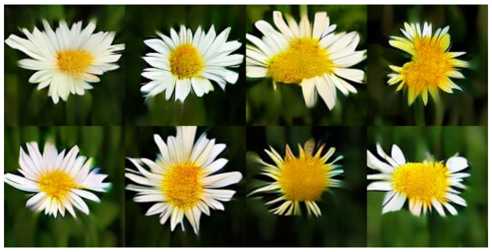

DiT-S/4 (Config A).

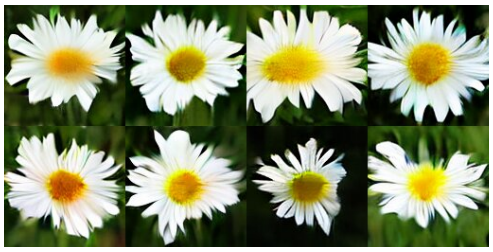

DiT-XS/2 (Config B).

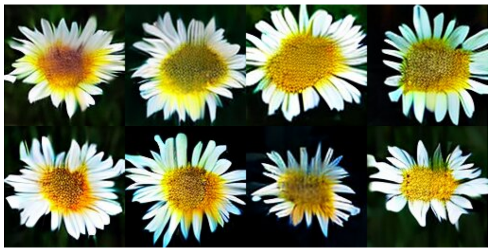

DiT-S/4 (Config B).


DiT-XS/2 (Config C).


DiT-S/4 (Config C).


DiT-XS/2 (Config D).


DiT-S/4 (Config D).


DiT-XS/2 (Config E).

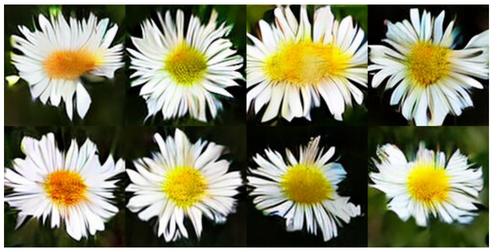

DiT-S/4 (Config E).

Figure 11: Uncurated samples of DiT-XS/2 and DiT-S/4 across all configurations.
Guidance scale = 5.0
EMA relative standard-deviation = 10 %
Class label = “Daisy” (985)

DiT-XS/2 (Config A).

DiT-S/4 (Config A).

DiT-XS/2 (Config B).

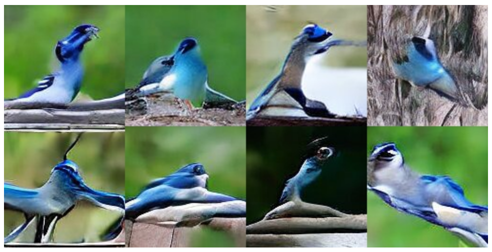
DiT-S/4 (Config B).

DiT-XS/2 (Config C).

DiT-S/4 (Config C).

DiT-XS/2 (Config D).

DiT-S/4 (Config D).


DiT-XS/2 (Config E).

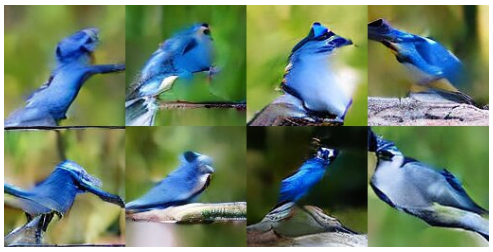

DiT-S/4 (Config E).

Figure 12: Uncurated samples of DiT-XS/2 and DiT-S/4 across all configurations.
Guidance scale = 5.0
EMA relative standard-deviation = $10\,\%$
Class label = "Jay" (17)

DiT-XS/2 (Config A).

DiT-S/4 (Config A).

DiT-XS/2 (Config B).

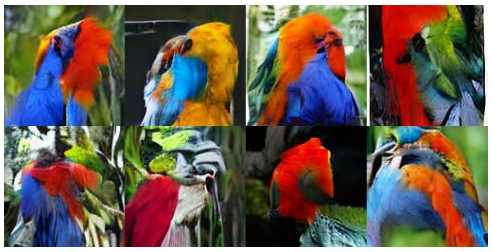
DiT-S/4 (Config B).

DiT-XS/2 (Config C).

DiT-S/4 (Config C).

DiT-XS/2 (Config D).

DiT-S/4 (Config D).


DiT-XS/2 (Config E).

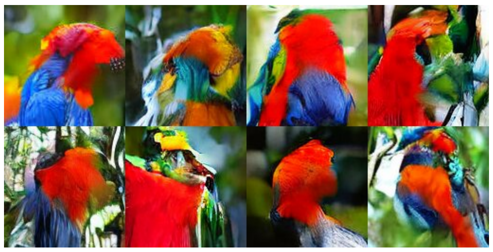

DiT-S/4 (Config E).

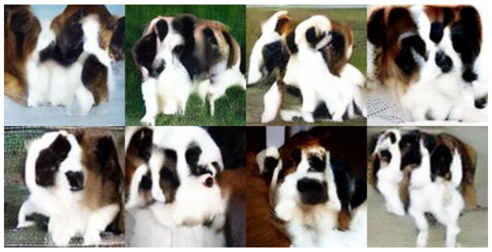
Figure 13: Uncurated samples of DiT-XS/2 and DiT-S/4 across all configurations.
Guidance scale = 5.0
EMA relative standard-deviation = 10 %
Class label = “Macaw” (88)

DiT-XS/2 (Config A).

DiT-S/4 (Config A).

DiT-XS/2 (Config B).

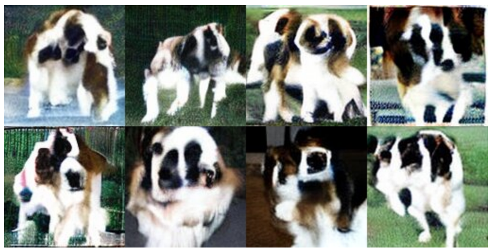
DiT-S/4 (Config B).


DiT-XS/2 (Config C).

DiT-S/4 (Config C).

DiT-XS/2 (Config D).

DiT-S/4 (Config D).


DiT-XS/2 (Config E).

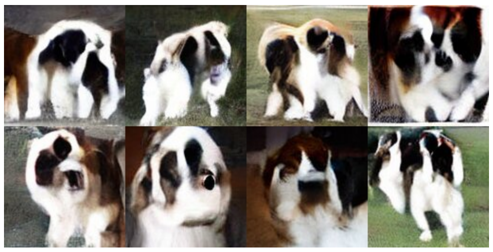

DiT-S/4 (Config E).

Figure 14: Uncurated samples of DiT-XS/2 and DiT-S/4 across all configurations.
Guidance scale = 5.0
EMA relative standard-deviation = $10\,\%$
Class label = “St. Bernard” (247)