arXiv:2212.09748v2 [cs.CV] 2 Mar 2023

Scalable Diffusion Models with Transformers

William Peebles*
UC Berkeley

Saining Xie
New York University

Figure 1. **Diffusion models with transformer backbones achieve state-of-the-art image quality.** We show selected samples from two of our class-conditional DiT-XL/2 models trained on ImageNet at 512×512 and 256×256 resolution, respectively.

Abstract

We explore a new class of diffusion models based on the transformer architecture. We train latent diffusion models of images, replacing the commonly-used U-Net backbone with a transformer that operates on latent patches. We analyze the scalability of our Diffusion Transformers (DiTs) through the lens of forward pass complexity as measured by Gflops. We find that DiTs with higher Gflops—through increased transformer depth/width or increased number of input tokens—consistently have lower FID. In addition to possessing good scalability properties, our largest DiT-XL/2 models outperform all prior diffusion models on the class-conditional ImageNet 512×512 and 256×256 benchmarks, achieving a state-of-the-art FID of 2.27 on the latter.

1. Introduction

Machine learning is experiencing a renaissance powered by transformers. Over the past five years, neural architectures for natural language processing [8, 42], vision [10] and several other domains have largely been subsumed by transformers [60]. Many classes of image-level generative models remain holdouts to the trend, though—while transformers see widespread use in autoregressive models [3,6,43,47], they have seen less adoption in other generative modeling frameworks. For example, diffusion models have been at the forefront of recent advances in image-level generative models [9,46]; yet, they all adopt a convolutional U-Net architecture as the de-facto choice of backbone.

* Work done during an internship at Meta AI, FAIR Team.
Code and project page available here.

Figure 2. **ImageNet generation with Diffusion Transformers (DiTs).** Bubble area indicates the flops of the diffusion model. *Left:* FID-50K (lower is better) of our DiT models at 400K training iterations. Performance steadily improves in FID as model flops increase. *Right:* Our best model, DiT-XL/2, is compute-efficient and outperforms all prior U-Net-based diffusion models, like ADM and LDM.

The seminal work of Ho *et al.* [19] first introduced the U-Net backbone for diffusion models. Having initially seen success within pixel-level autoregressive models and conditional GANs [23], the U-Net was inherited from PixelCNN++ [52, 58] with a few changes. The model is convolutional, comprised primarily of ResNet [15] blocks. In contrast to the standard U-Net [49], additional spatial self-attention blocks, which are essential components in transformers, are interspersed at lower resolutions. Dhariwal and Nichol [9] ablated several architecture choices for the U-Net, such as the use of adaptive normalization layers [40] to inject conditional information and channel counts for convolutional layers. However, the high-level design of the U-Net from Ho *et al.* has largely remained intact.

With this work, we aim to demystify the significance of architectural choices in diffusion models and offer empirical baselines for future generative modeling research. We show that the U-Net inductive bias is *not* crucial to the performance of diffusion models, and they can be readily replaced with standard designs such as transformers. As a result, diffusion models are well-poised to benefit from the recent trend of architecture unification—e.g., by inheriting best practices and training recipes from other domains, as well as retaining favorable properties like scalability, robustness and efficiency. A standardized architecture would also open up new possibilities for cross-domain research.

In this paper, we focus on a new class of diffusion models based on transformers. We call them *Diffusion Transformers*, or DiTs for short. DiTs adhere to the best practices of Vision Transformers (ViTs) [10], which have been shown to scale more effectively for visual recognition than traditional convolutional networks (e.g., ResNet [15]).

More specifically, we study the scaling behavior of transformers with respect to *network complexity* vs. *sample quality*. We show that by constructing and benchmarking the DiT design space under the *Latent Diffusion Models* (LDMs) [48] framework, where diffusion models are trained within a VAE's latent space, we can successfully replace the U-Net backbone with a transformer. We further show that DiTs are scalable architectures for diffusion models: there is a strong correlation between the network complexity (measured by Gflops) *vs.* sample quality (measured by FID). By simply scaling-up DiT and training an LDM with a high-capacity backbone (118.6 Gflops), we are able to achieve a state-of-the-art result of 2.27 FID on the class-conditional 256×256 ImageNet generation benchmark.

2. Related Work

Transformers. Transformers [60] have replaced domain-specific architectures across language, vision [10], reinforcement learning [5, 25] and meta-learning [39]. They have shown remarkable scaling properties under increasing model size, training compute and data in the language domain [26], as generic autoregressive models [17] and as ViTs [63]. Beyond language, transformers have been trained to autoregressively predict pixels [6, 7, 38]. They have also been trained on discrete codebooks [59] as both autoregressive models [11, 47] and masked generative models [4, 14]; the former has shown excellent scaling behavior up to 20B parameters [62]. Finally, transformers have been explored in DDPMs to synthesize non-spatial data; e.g., to generate CLIP image embeddings in DALL·E 2 [41, 46]. In this paper, we study the scaling properties of transformers when used as the backbone of diffusion models of images.

Figure 3. **The Diffusion Transformer (DiT) architecture.** *Left:* We train conditional latent DiT models. The input latent is decomposed into patches and processed by several DiT blocks. *Right:* Details of our DiT blocks. We experiment with variants of standard transformer blocks that incorporate conditioning via adaptive layer norm, cross-attention and extra input tokens. Adaptive layer norm works best.

Denoising diffusion probabilistic models (DDPMs). Diffusion [19, 54] and score-based generative models [22, 56] have been particularly successful as generative models of images [35,46,48,50], in many cases outperforming generative adversarial networks (GANs) [12] which had previously been state-of-the-art. Improvements in DDPMs over the past two years have largely been driven by improved sampling techniques [19, 27, 55], most notably classifier-free guidance [21], reformulating diffusion models to predict noise instead of pixels [19] and using cascaded DDPM pipelines where low-resolution base diffusion models are trained in parallel with upsamplers [9, 20]. For all the diffusion models listed above, convolutional U-Nets [49] are the de-facto choice of backbone architecture. Concurrent work [24] introduced a novel, efficient architecture based on attention for DDPMs; we explore pure transformers.

Architecture complexity. When evaluating architecture complexity in the image generation literature, it is fairly common practice to use parameter counts. In general, parameter counts can be poor proxies for the complexity of image models since they do not account for, e.g., image resolution which significantly impacts performance [44, 45]. Instead, much of the model complexity analysis in this paper is through the lens of theoretical Gflops. This brings us in-line with the architecture design literature where Gflops are widely-used to gauge complexity. In practice, the golden complexity metric is still up for debate as it frequently depends on particular application scenarios. Nichol and Dhariwal's seminal work improving diffusion models [9, 36] is most related to us—there, they analyzed the scalability and Gflop properties of the U-Net architecture class. In this paper, we focus on the transformer class.

3. Diffusion Transformers

3.1. Preliminaries

Diffusion formulation. Before introducing our architecture, we briefly review some basic concepts needed to understand diffusion models (DDPMs) [19, 54]. Gaussian diffusion models assume a forward noising process which gradually applies noise to real data x_0: $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$, where constants $\bar{\alpha}_t$ are hyperparameters. By applying the reparameterization trick, we can sample $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, \mathbf{I})$.

Diffusion models are trained to learn the reverse process that inverts forward process corruptions: $p_\theta(x_{t-1}|x_t) = \mathcal{N}(\mu_\theta(x_t), \Sigma_\theta(x_t))$, where neural networks are used to predict the statistics of p_θ. The reverse process model is trained with the variational lower bound [30] of the log-likelihood of x_0, which reduces to $\mathcal{L}(\theta) = -p(x_0|x_1) + \sum_t \mathcal{D}_{KL}(q^*(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))$, excluding an additional term irrelevant for training. Since both q^* and p_θ are Gaussian, $\mathcal{D}_{KL}$ can be evaluated with the mean and covariance of the two distributions. By reparameterizing μ_θ as a noise prediction network ϵ_θ, the model can be trained using simple mean-squared error between the predicted noise $\epsilon_\theta(x_t)$ and the ground truth sampled Gaussian noise ϵ_t: $\mathcal{L}_{simple}(\theta) = ||\epsilon_\theta(x_t) - \epsilon_t||_2^2$. But, in order to train diffusion models with a learned reverse process covariance Σ_θ, the full $\mathcal{D}_{KL}$ term needs to be optimized. We follow Nichol and Dhariwal's approach [36]: train ϵ_θ with $\mathcal{L}_{simple}$, and train Σ_θ with the full $\mathcal{L}$. Once p_θ is trained, new images can be sampled by initializing $x_{t_{\max}} \sim \mathcal{N}(0, \mathbf{I})$ and sampling $x_{t-1} \sim p_\theta(x_{t-1}|x_t)$ via the reparameterization trick.

Classifier-free guidance. Conditional diffusion models take extra information as input, such as a class label c. In this case, the reverse process becomes $p_\theta(x_{t-1}|x_t, c)$, where ϵ_θ and Σ_θ are conditioned on c. In this setting, *classifier-free guidance* can be used to encourage the sampling procedure to find x such that $\log p(c|x)$ is high [21]. By Bayes Rule, $\log p(c|x) \propto \log p(x|c) - \log p(x)$, and hence $\nabla_x \log p(c|x) \propto \nabla_x \log p(x|c) - \nabla_x \log p(x)$. By interpreting the output of diffusion models as the score function, the DDPM sampling procedure can be guided to sample x with high $p(x|c)$ by: $\hat{\epsilon}_\theta(x_t, c) = \epsilon_\theta(x_t, \emptyset) + s \cdot \nabla_x \log p(x|c) \propto \epsilon_\theta(x_t, \emptyset) + s \cdot (\epsilon_\theta(x_t, c) - \epsilon_\theta(x_t, \emptyset))$, where $s > 1$ indicates the scale of the guidance (note that $s = 1$ recovers standard sampling). Evaluating the diffusion model with $c = \emptyset$ is done by randomly dropping out c during training and replacing it with a learned "null" embedding $\emptyset$. Classifier-free guidance is widely-known to yield significantly improved samples over generic sampling techniques [21, 35, 46], and the trend holds for our DiT models.

Latent diffusion models. Training diffusion models directly in high-resolution pixel space can be computationally prohibitive. *Latent diffusion models* (LDMs) [48] tackle this issue with a two-stage approach: (1) learn an autoencoder that compresses images into smaller spatial representations with a learned encoder E; (2) train a diffusion model of representations $z = E(x)$ instead of a diffusion model of images x (E is frozen). New images can then be generated by sampling a representation z from the diffusion model and subsequently decoding it to an image with the learned decoder $x = D(z)$.

As shown in Figure 2, LDMs achieve good performance while using a fraction of the Gflops of pixel space diffusion models like ADM. Since we are concerned with compute efficiency, this makes them an appealing starting point for architecture exploration. In this paper, we apply DiTs to latent space, although they could be applied to pixel space without modification as well. This makes our image generation pipeline a hybrid-based approach; we use off-the-shelf convolutional VAEs and transformer-based DDPMs.

3.2. Diffusion Transformer Design Space

We introduce Diffusion Transformers (DiTs), a new architecture for diffusion models. We aim to be as faithful to the standard transformer architecture as possible to retain its scaling properties. Since our focus is training DDPMs of images (specifically, spatial representations of images), DiT is based on the Vision Transformer (ViT) architecture which operates on sequences of patches [10]. DiT retains many of the best practices of ViTs. Figure 3 shows an overview of the complete DiT architecture. In this section, we describe the forward pass of DiT, as well as the components of the design space of the DiT class.

Figure 4. **Input specifications for DiT.** Given patch size $p \times p$, a spatial representation (the noised latent from the VAE) of shape $I \times I \times C$ is "patchified" into a sequence of length $T = (I/p)^2$ with hidden dimension d. A smaller patch size p results in a longer sequence length and thus more Gflops.

Patchify. The input to DiT is a spatial representation z (for $256 \times 256 \times 3$ images, z has shape $32 \times 32 \times 4$). The first layer of DiT is "patchify," which converts the spatial input into a sequence of T tokens, each of dimension d, by linearly embedding each patch in the input. Following patchify, we apply standard ViT frequency-based positional embeddings (the sine-cosine version) to all input tokens. The number of tokens T created by patchify is determined by the patch size hyperparameter p. As shown in Figure 4, halving p will quadruple T, and thus *at least* quadruple total transformer Gflops. Although it has a significant impact on Gflops, note that changing p has no meaningful impact on downstream parameter counts.

We add $p = 2, 4, 8$ to the DiT design space.

DiT block design. Following patchify, the input tokens are processed by a sequence of transformer blocks. In addition to noised image inputs, diffusion models sometimes process additional conditional information such as noise timesteps t, class labels c, natural language, etc. We explore four variants of transformer blocks that process conditional inputs differently. The designs introduce small, but important, modifications to the standard ViT block design. The designs of all blocks are shown in Figure 3.

- *In-context conditioning.* We simply append the vector embeddings of t and c as two additional tokens in the input sequence, treating them no differently from the image tokens. This is similar to `cls` tokens in ViTs, and it allows us to use standard ViT blocks without modification. After the final block, we remove the conditioning tokens from the sequence. This approach introduces negligible new Gflops to the model.

Figure 5. **Comparing different conditioning strategies.** adaLN-Zero outperforms cross-attention and in-context conditioning at all stages of training.

– *Cross-attention block*. We concatenate the embeddings of t and c into a length-two sequence, separate from the image token sequence. The transformer block is modified to include an additional multi-head cross-attention layer following the multi-head self-attention block, similar to the original design from Vaswani *et al.* [60], and also similar to the one used by LDM for conditioning on class labels. Cross-attention adds the most Gflops to the model, roughly a 15% overhead.

– *Adaptive layer norm (adaLN) block*. Following the widespread usage of adaptive normalization layers [40] in GANs [2, 28] and diffusion models with U-Net backbones [9], we explore replacing standard layer norm layers in transformer blocks with adaptive layer norm (adaLN). Rather than directly learn dimension-wise scale and shift parameters γ and β, we regress them from the sum of the embedding vectors of t and c. Of the three block designs we explore, adaLN adds the least Gflops and is thus the most compute-efficient. It is also the only conditioning mechanism that is restricted to apply the *same function* to all tokens.

– *adaLN-Zero block*. Prior work on ResNets has found that initializing each residual block as the identity function is beneficial. For example, Goyal *et al.* found that zero-initializing the final batch norm scale factor γ in each block accelerates large-scale training in the supervised learning setting [13]. Diffusion U-Net models use a similar initialization strategy, zero-initializing the final convolutional layer in each block prior to any residual connections. We explore a modification of the adaLN DiT block which does the same. In addition to regressing γ and β, we also regress dimension-wise scaling parameters α that are applied immediately prior to any residual connections within the DiT block.

Model	Layers N	Hidden size d	Heads	Gflops $_{(I=32,\ p=4)}$
DiT-S	12	384	6	1.4
DiT-B	12	768	12	5.6
DiT-L	24	1024	16	19.7
DiT-XL	28	1152	16	29.1

Table 1. **Details of DiT models.** We follow ViT [10] model configurations for the Small (S), Base (B) and Large (L) variants; we also introduce an XLarge (XL) config as our largest model.

We initialize the MLP to output the zero-vector for all α; this initializes the full DiT block as the identity function. As with the vanilla adaLN block, adaLN-Zero adds negligible Gflops to the model.

We include the in-context, cross-attention, adaptive layer norm and adaLN-Zero blocks in the DiT design space.

Model size. We apply a sequence of N DiT blocks, each operating at the hidden dimension size d. Following ViT, we use standard transformer configs that jointly scale N, d and attention heads [10, 63]. Specifically, we use four configs: DiT-S, DiT-B, DiT-L and DiT-XL. They cover a wide range of model sizes and flop allocations, from 0.3 to 118.6 Gflops, allowing us to gauge scaling performance. Table 1 gives details of the configs.

We add B, S, L and XL configs to the DiT design space.

Transformer decoder. After the final DiT block, we need to decode our sequence of image tokens into an output noise prediction and an output diagonal covariance prediction. Both of these outputs have shape equal to the original spatial input. We use a standard linear decoder to do this; we apply the final layer norm (adaptive if using adaLN) and linearly decode each token into a $p \times p \times 2C$ tensor, where C is the number of channels in the spatial input to DiT. Finally, we rearrange the decoded tokens into their original spatial layout to get the predicted noise and covariance.

The complete DiT design space we explore is patch size, transformer block architecture and model size.

4. Experimental Setup

We explore the DiT design space and study the scaling properties of our model class. Our models are named according to their configs and latent patch sizes p; for example, DiT-XL/2 refers to the XLarge config and $p = 2$.

Training. We train class-conditional latent DiT models at 256×256 and 512×512 image resolution on the ImageNet dataset [31], a highly-competitive generative modeling benchmark. We initialize the final linear layer with zeros and otherwise use standard weight initialization techniques from ViT. We train all models with AdamW [29, 33].

Figure 6. **Scaling the DiT model improves FID at all stages of training.** We show FID-50K over training iterations for 12 of our DiT models. *Top row:* We compare FID holding patch size constant. *Bottom row:* We compare FID holding model size constant. Scaling the transformer backbone yields better generative models across all model sizes and patch sizes.

We use a constant learning rate of 1×10^{-4}, no weight decay and a batch size of 256. The only data augmentation we use is horizontal flips. Unlike much prior work with ViTs [57, 61], we did not find learning rate warmup nor regularization necessary to train DiTs to high performance. Even without these techniques, training was highly stable across all model configs and we did not observe any loss spikes commonly seen when training transformers. Following common practice in the generative modeling literature, we maintain an exponential moving average (EMA) of DiT weights over training with a decay of 0.9999. All results reported use the EMA model. We use identical training hyperparameters across all DiT model sizes and patch sizes. Our training hyperparameters are almost entirely retained from ADM. *We did not tune learning rates, decay/warm-up schedules, Adam β_1/β_2 or weight decays.*

Diffusion. We use an off-the-shelf pre-trained variational autoencoder (VAE) model [30] from Stable Diffusion [48]. The VAE encoder has a downsample factor of 8—given an RGB image x with shape $256 \times 256 \times 3$, $z = E(x)$ has shape $32 \times 32 \times 4$. Across all experiments in this section, our diffusion models operate in this $\mathcal{Z}$-space. After sampling a new latent from our diffusion model, we decode it to pixels using the VAE decoder $x = D(z)$. We retain diffusion hyperparameters from ADM [9]; specifically, we use a $t_{\text{max}} = 1000$ linear variance schedule ranging from 1×10^{-4} to 2×10^{-2}, ADM's parameterization of the covariance Σ_θ and their method for embedding input timesteps and labels.

Evaluation metrics. We measure scaling performance with Fréchet Inception Distance (FID) [18], the standard metric for evaluating generative models of images. We follow convention when comparing against prior works and report FID-50K using 250 DDPM sampling steps. FID is known to be sensitive to small implementation details [37]; to ensure accurate comparisons, all values reported in this paper are obtained by exporting samples and using ADM's TensorFlow evaluation suite [9]. FID numbers reported in this section do *not* use classifier-free guidance except where otherwise stated. We additionally report Inception Score [51], sFID [34] and Precision/Recall [32] as secondary metrics.

Compute. We implement all models in JAX [1] and train them using TPU-v3 pods. DiT-XL/2, our most compute-intensive model, trains at roughly 5.7 iterations/second on a TPU v3-256 pod with a global batch size of 256.

5. Experiments

DiT block design. We train four of our highest Gflop DiT-XL/2 models, each using a different block design—in-context (119.4 Gflops), cross-attention (137.6 Gflops), adaptive layer norm (adaLN, 118.6 Gflops) or adaLN-zero (118.6 Gflops). We measure FID over the course of training. Figure 5 shows the results. The adaLN-Zero block yields lower FID than both cross-attention and in-context conditioning while being the most compute-efficient. At 400K training iterations, the FID achieved with the adaLN-Zero model is nearly half that of the in-context model, demonstrating that the conditioning mechanism critically affects model quality. Initialization is also important—adaLN-Zero, which initializes each DiT block as the identity function, significantly outperforms vanilla adaLN. *For the rest of the paper, all models will use adaLN-Zero DiT blocks.*

Figure 7. **Increasing transformer forward pass Gflops increases sample quality.** *Best viewed zoomed-in.* We sample from all 12 of our DiT models after 400K training steps using the same input latent noise and class label. Increasing the Gflops in the model—either by increasing transformer depth/width or increasing the number of input tokens—yields significant improvements in visual fidelity.

Figure 8. **Transformer Gflops are strongly correlated with FID.** We plot the Gflops of each of our DiT models and each model's FID-50K after 400K training steps.

Figure 9. **Larger DiT models use large compute more efficiently.** We plot FID as a function of total training compute.

Scaling model size and patch size. We train 12 DiT models, sweeping over model configs (S, B, L, XL) and patch sizes (8, 4, 2). Note that DiT-L and DiT-XL are significantly closer to each other in terms of relative Gflops than other configs. Figure 2 (left) gives an overview of the Gflops of each model and their FID at 400K training iterations. In all cases, we find that increasing model size and decreasing patch size yields considerably improved diffusion models.

Figure 6 (top) demonstrates how FID changes as model size is increased and patch size is held constant. Across all four configs, significant improvements in FID are obtained over all stages of training by making the transformer deeper and wider. Similarly, Figure 6 (bottom) shows FID as patch size is decreased and model size is held constant. We again observe considerable FID improvements throughout training by simply scaling the number of tokens processed by DiT, holding parameters approximately fixed.

DiT Gflops are critical to improving performance. The results of Figure 6 suggest that parameter counts do not uniquely determine the quality of a DiT model. As model size is held constant and patch size is decreased, the transformer's total parameters are effectively unchanged (actually, total parameters slightly *decrease*), and only Gflops are increased. These results indicate that scaling model *Gflops* is actually the key to improved performance. To investigate this further, we plot the FID-50K at 400K training steps against model Gflops in Figure 8. The results demonstrate that different DiT configs obtain similar FID values when their total Gflops are similar (e.g., DiT-S/2 and DiT-B/4). We find a strong negative correlation between model Gflops and FID-50K, suggesting that additional model compute is the critical ingredient for improved DiT models. In Figure 12 (appendix), we find that this trend holds for other metrics such as Inception Score.

Larger DiT models are more compute-efficient. In Figure 9, we plot FID as a function of total training compute for all DiT models. We estimate training compute as model Gflops · batch size · training steps · 3, where the factor of 3 roughly approximates the backwards pass as being twice as compute-heavy as the forward pass. We find that small DiT models, even when trained longer, eventually become compute-inefficient relative to larger DiT models trained for fewer steps. Similarly, we find that models that are identical except for patch size have different performance profiles even when controlling for training Gflops. For example, XL/4 is outperformed by XL/2 after roughly 10^{10} Gflops.

Visualizing scaling. We visualize the effect of scaling on sample quality in Figure 7. At 400K training steps, we sample an image from each of our 12 DiT models using *identical* starting noise $x_{t_{\max}}$, sampling noise and class labels. This lets us visually interpret how scaling affects DiT sample quality. Indeed, scaling both model size and the number of tokens yields notable improvements in visual quality.

5.1. State-of-the-Art Diffusion Models

256×256 ImageNet. Following our scaling analysis, we continue training our highest Gflop model, DiT-XL/2, for 7M steps. We show samples from the model in Figures 1, and we compare against state-of-the-art class-conditional generative models. We report results in Table 2. When using classifier-free guidance, DiT-XL/2 outperforms all prior diffusion models, decreasing the previous best FID-50K of 3.60 achieved by LDM to 2.27. Figure 2 (right) shows that DiT-XL/2 (118.6 Gflops) is compute-efficient relative to latent space U-Net models like LDM-4 (103.6 Gflops) and substantially more efficient than pixel space U-Net models such as ADM (1120 Gflops) or ADM-U (742 Gflops).

Class-Conditional ImageNet 256×256					
Model	FID↓	sFID↓	IS↑	Precision↑	Recall↑
BigGAN-deep [2]	6.95	7.36	171.4	0.87	0.28
StyleGAN-XL [53]	2.30	4.02	265.12	0.78	0.53
ADM [9]	10.94	6.02	100.98	0.69	0.63
ADM-U	7.49	5.13	127.49	0.72	0.63
ADM-G	4.59	5.25	186.70	0.82	0.52
ADM-G, ADM-U	3.94	6.14	215.84	0.83	0.53
CDM [20]	4.88	-	158.71	-	-
LDM-8 [48]	15.51	-	79.03	0.65	0.63
LDM-8-G	7.76	-	209.52	0.84	0.35
LDM-4	10.56	-	103.49	0.71	0.62
LDM-4-G (cfg=1.25)	3.95	-	178.22	0.81	0.55
LDM-4-G (cfg=1.50)	3.60	-	247.67	**0.87**	0.48
DiT-XL/2	9.62	6.85	121.50	0.67	**0.67**
DiT-XL/2-G (cfg=1.25)	3.22	5.28	201.77	0.76	0.62
DiT-XL/2-G (cfg=1.50)	**2.27**	**4.60**	**278.24**	0.83	0.57

Table 2. **Benchmarking class-conditional image generation on ImageNet 256×256.** DiT-XL/2 achieves state-of-the-art FID.

Class-Conditional ImageNet 512×512					
Model	FID↓	sFID↓	IS↑	Precision↑	Recall↑
BigGAN-deep [2]	8.43	8.13	177.90	0.88	0.29
StyleGAN-XL [53]	2.41	4.06	267.75	0.77	0.52
ADM [9]	23.24	10.19	58.06	0.73	0.60
ADM-U	9.96	5.62	121.78	0.75	**0.64**
ADM-G	7.72	6.57	172.71	**0.87**	0.42
ADM-G, ADM-U	3.85	5.86	221.72	0.84	0.53
DiT-XL/2	12.03	7.12	105.25	0.75	**0.64**
DiT-XL/2-G (cfg=1.25)	4.64	5.77	174.77	0.81	0.57
DiT-XL/2-G (cfg=1.50)	**3.04**	**5.02**	**240.82**	0.84	0.54

Table 3. **Benchmarking class-conditional image generation on ImageNet 512×512.** Note that prior work [9] measures Precision and Recall using 1000 real samples for 512 × 512 resolution; for consistency, we do the same.

Our method achieves the lowest FID of all prior generative models, including the previous state-of-the-art StyleGAN-XL [53]. Finally, we also observe that DiT-XL/2 achieves higher recall values at all tested classifier-free guidance scales compared to LDM-4 and LDM-8. When trained for only 2.35M steps (similar to ADM), XL/2 still outperforms all prior diffusion models with an FID of 2.55.

512×512 ImageNet. We train a new DiT-XL/2 model on ImageNet at 512 × 512 resolution for 3M iterations with identical hyperparameters as the 256 × 256 model. With a patch size of 2, this XL/2 model processes a total of 1024 tokens after patchifying the 64 × 64 × 4 input latent (524.6 Gflops). Table 3 shows comparisons against state-of-the-art methods. XL/2 again outperforms all prior diffusion models at this resolution, improving the previous best FID of 3.85 achieved by ADM to 3.04. Even with the increased number of tokens, XL/2 remains compute-efficient. For example, ADM uses 1983 Gflops and ADM-U uses 2813 Gflops; XL/2 uses 524.6 Gflops. We show samples from the high-resolution XL/2 model in Figure 1 and the appendix.

Figure 10. **Scaling-up *sampling* compute does not compensate for a lack of *model* compute.** For each of our DiT models trained for 400K iterations, we compute FID-10K using [16, 32, 64, 128, 256, 1000] sampling steps. For each number of steps, we plot the FID as well as the Gflops used to sample each image. Small models cannot close the performance gap with our large models, even if they sample with more test-time Gflops than the large models.

5.2. Scaling Model vs. Sampling Compute

Diffusion models are unique in that they can use additional compute after training by increasing the number of sampling steps when generating an image. Given the impact of model Gflops on sample quality, in this section we study if smaller-*model compute* DiTs can outperform larger ones by using more *sampling compute*. We compute FID for all 12 of our DiT models after 400K training steps, using [16, 32, 64, 128, 256, 1000] sampling steps per-image. The main results are in Figure 10. Consider DiT-L/2 using 1000 sampling steps versus DiT-XL/2 using 128 steps. In this case, L/2 uses 80.7 Tflops to sample each image; XL/2 uses 5× less compute—15.2 Tflops—to sample each image. Nonetheless, XL/2 has the better FID-10K (23.7 vs 25.9). In general, scaling-up sampling compute *cannot* compensate for a lack of model compute.

6. Conclusion

We introduce Diffusion Transformers (DiTs), a simple transformer-based backbone for diffusion models that outperforms prior U-Net models and inherits the excellent scaling properties of the transformer model class. Given the promising scaling results in this paper, future work should continue to scale DiTs to larger models and token counts. DiT could also be explored as a drop-in backbone for text-to-image models like DALL·E 2 and Stable Diffusion.

Acknowledgements. We thank Kaiming He, Ronghang Hu, Alexander Berg, Shoubhik Debnath, Tim Brooks, Ilija Radosavovic and Tete Xiao for helpful discussions. William Peebles is supported by the NSF GRFP.

References

[1] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. 6

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019. 5, 9

[3] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020. 1

[4] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *CVPR*, pages 11315–11325, 2022. 2

[5] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, 2021. 2

[6] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020. 1, 2

[7] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 2

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HCT*, 2019. 1

[9] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021. 1, 2, 3, 5, 6, 9, 12

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020. 1, 2, 4, 5

[11] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2020. 2

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 3

[13] 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

[14] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *CVPR*, pages 10696–10706, 2022. 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[16] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 12

[17] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020. 2

[18] 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. 2017. 6

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 2, 3

[20] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *arXiv:2106.15282*, 2021. 3, 9

[21] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. 3, 4

[22] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005. 3

[23] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 2

[24] Allan Jabri, David Fleet, and Ting Chen. Scalable adaptive computation for iterative generation. *arXiv preprint arXiv:2212.11972*, 2022. 3

[25] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *NeurIPS*, 2021. 2

[26] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv:2001.08361*, 2020. 2, 13

[27] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Proc. NeurIPS*, 2022. 3

[28] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 5

[29] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[30] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 3, 6

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 5

[32] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *NeurIPS*, 2019. 6

[33] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv:1711.05101*, 2017. 5

[34] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W Battaglia. Generating images with sparse representations. *arXiv preprint arXiv:2103.03841*, 2021. 6

[35] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv:2112.10741*, 2021. 3, 4

[36] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021. 3

[37] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *CVPR*, 2022. 6

[38] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International conference on machine learning*, pages 4055–4064. PMLR, 2018. 2

[39] William Peebles, Ilija Radosavovic, Tim Brooks, Alexei Efros, and Jitendra Malik. Learning to learn with generative models of neural network checkpoints. *arXiv preprint arXiv:2209.12892*, 2022. 2

[40] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018. 2, 5

[41] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 2

[42] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. 1

[43] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. 2019. 1

[44] Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollár. On network design spaces for visual recognition. In *ICCV*, 2019. 3

[45] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, 2020. 3

[46] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv:2204.06125*, 2022. 1, 2, 3, 4

[47] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021. 1, 2

[48] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2, 3, 4, 6, 9

[49] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2, 3

[50] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv:2205.11487*, 2022. 3

[51] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training GANs. In *NeurIPS*, 2016. 6

[52] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. PixelCNN++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017. 2

[53] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *SIGGRAPH*, 2022. 9

[54] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. 3

[55] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv:2010.02502*, 2020. 3

[56] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019. 3

[57] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your ViT? data, augmentation, and regularization in vision transformers. *TMLR*, 2022. 6

[58] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016. 2

[59] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 2

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1, 2, 5

[61] Tete Xiao, Piotr Dollar, Mannat Singh, Eric Mintun, Trevor Darrell, and Ross Girshick. Early convolutions help transformers see better. In *NeurIPS*, 2021. 6

[62] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv:2206.10789*, 2022. 2

[63] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *CVPR*, 2022. 2, 5

Figure 11. **Additional selected samples from our 512×512 and 256×256 resolution DiT-XL/2 models.** We use a classifier-free guidance scale of 6.0 for the 512×512 model and 4.0 for the 256×256 model. Both models use the ft-EMA VAE decoder.

A. Additional Implementation Details

We include detailed information about all of our DiT models in Table 4, including both 256×256 and 512×512 models. In Figure 13, we report DiT training loss curves. Finally, we also include Gflop counts for DDPM U-Net models from ADM and LDM in Table 6.

DiT model details. To embed input timesteps, we use a 256-dimensional frequency embedding [9] followed by a two-layer MLP with dimensionality equal to the transformer's hidden size and SiLU activations. Each adaLN layer feeds the sum of the timestep and class embeddings into a SiLU nonlinearity and a linear layer with output neurons equal to either $4\times$ (adaLN) or $6\times$ (adaLN-Zero) the transformer's hidden size. We use GELU nonlinearities (approximated with tanh) in the core transformer [16].

Classifier-free guidance on a subset of channels. In our experiments using classifier-free guidance, we applied guidance only to the first three channels of the latents instead of all four channels. Upon investigating, we found that three-channel guidance and four-channel guidance give similar results (in terms of FID) when simply adjusting the scale factor. Specifically, three-channel guidance with a scale of $(1 + x)$ appears reasonably well-approximated by four-channel guidance with a scale of $(1 + \frac{3}{4}x)$ (e.g., three-channel guidance with a scale of 1.5 gives an FID-50K of 2.27, and four-channel guidance with a scale of 1.375 gives an FID-50K of 2.20). It is somewhat interesting that applying guidance to a subset of elements can still yield good performance, and we leave it to future work to explore this phenomenon further.

B. Model Samples

We show samples from our two DiT-XL/2 models at 512×512 and 256×256 resolution trained for 3M and 7M steps, respectively. Figures 1 and 11 show selected samples from both models. Figures 14 through 33 show *uncurated* samples from the two models across a range of classifier-free guidance scales and input class labels (generated with 250 DDPM sampling steps and the ft-EMA VAE decoder). As with prior work using guidance, we observe that larger scales increase visual fidelity and decrease sample diversity.

Model	Image Resolution	Flops (G)	Params (M)	Training Steps (K)	Batch Size	Learning Rate	DiT Block	FID-50K (no guidance)
DiT-S/8	256×256	0.36	33	400	256	1×10^{-4}	adaLN-Zero	153.60
DiT-S/4	256×256	1.41	33	400	256	1×10^{-4}	adaLN-Zero	100.41
DiT-S/2	256×256	6.06	33	400	256	1×10^{-4}	adaLN-Zero	68.40
DiT-B/8	256×256	1.42	131	400	256	1×10^{-4}	adaLN-Zero	122.74
DiT-B/4	256×256	5.56	130	400	256	1×10^{-4}	adaLN-Zero	68.38
DiT-B/2	256×256	23.01	130	400	256	1×10^{-4}	adaLN-Zero	43.47
DiT-L/8	256×256	5.01	459	400	256	1×10^{-4}	adaLN-Zero	118.87
DiT-L/4	256×256	19.70	458	400	256	1×10^{-4}	adaLN-Zero	45.64
DiT-L/2	256×256	80.71	458	400	256	1×10^{-4}	adaLN-Zero	23.33
DiT-XL/8	256×256	7.39	676	400	256	1×10^{-4}	adaLN-Zero	106.41
DiT-XL/4	256×256	29.05	675	400	256	1×10^{-4}	adaLN-Zero	43.01
DiT-XL/2	256×256	118.64	675	400	256	1×10^{-4}	adaLN-Zero	19.47
DiT-XL/2	256×256	119.37	449	400	256	1×10^{-4}	in-context	35.24
DiT-XL/2	256×256	137.62	598	400	256	1×10^{-4}	cross-attention	26.14
DiT-XL/2	256×256	118.56	600	400	256	1×10^{-4}	adaLN	25.21
DiT-XL/2	256×256	118.64	675	2352	256	1×10^{-4}	adaLN-Zero	10.67
DiT-XL/2	256×256	118.64	675	7000	256	1×10^{-4}	adaLN-Zero	9.62
DiT-XL/2	512×512	524.60	675	1301	256	1×10^{-4}	adaLN-Zero	13.78
DiT-XL/2	512×512	524.60	675	3000	256	1×10^{-4}	adaLN-Zero	11.93

Table 4. **Details of all DiT models.** We report detailed information about every DiT model in our paper. Note that FID-50K here is computed *without* classifier-free guidance. Parameter and flop counts exclude the VAE model which contains 84M parameters across the encoder and decoder. For both the 256×256 and 512×512 DiT-XL/2 models, we never observed FID saturate and continued training them as long as possible. Numbers reported in this table use the ft-MSE VAE decoder.

C. Additional Scaling Results

Impact of scaling on metrics beyond FID. In Figure 12, we show the effects of DiT scale on a suite of evaluation metrics—FID, sFID, Inception Score, Precision and Recall. We find that our FID-driven analysis in the main paper generalizes to the other metrics—across every metric, scaled-up DiT models are more compute-efficient and model Gflops are highly-correlated with performance. In particular, Inception Score and Precision benefit heavily from increased model scale.

Impact of scaling on training loss. We also examine the impact of scale on training loss in Figure 13. Increasing DiT model Gflops (via transformer size or number of input tokens) causes the training loss to decrease more rapidly and saturate at a lower value. This phenomenon is consistent with trends observed with language models, where scaled-up transformers demonstrate both improved loss curves as well as improved performance on downstream evaluation suites [26].

D. VAE Decoder Ablations

We used off-the-shelf, pre-trained VAEs across our experiments. The VAE models (ft-MSE and ft-EMA) are fine-tuned versions of the original LDM "f8" model (only the decoder weights are fine-tuned). We monitored metrics for our scaling analysis in Section 5 using the ft-MSE decoder, and we used the ft-EMA decoder for our final metrics reported in Tables 2 and 3. In this section, we ablate three different choices of the VAE decoder; the original one used by LDM and the two fine-tuned decoders used by Stable Diffusion. Because the encoders are identical across models, the decoders can be swapped-in without retraining the diffusion model. Table 5 shows results; XL/2 continues to outperform all prior diffusion models when using the LDM decoder.

Class-Conditional ImageNet 256×256, DiT-XL/2-G (cfg=1.5)					
Decoder	FID↓	sFID↓	IS↑	Precision↑	Recall↑
original	2.46	5.18	271.56	0.82	0.57
ft-MSE	2.30	4.73	276.09	0.83	0.57
ft-EMA	2.27	4.60	278.24	0.83	0.57

Table 5. **Decoder ablation.** We tested different pre-trained VAE decoder weights available at `https://huggingface.co/stabilityai/sd-vae-ft-mse`. Different pre-trained decoder weights yield comparable results on ImageNet 256×256.

Diffusion U-Net Model Complexities				
Model	Image Resolution	Base Flops (G)	Upsampler Flops (G)	Total Flops (G)
ADM	128×128	307	-	307
ADM	256×256	1120	-	1120
ADM	512×512	1983	-	1983
ADM-U	256×256	110	632	742
ADM-U	512×512	307	2506	2813
LDM-4	256×256	104	-	104
LDM-8	256×256	57	-	57

Table 6. **Gflop counts for baseline diffusion models that use U-Net backbones.** Note that we only count Flops for DDPM components.

Figure 12. **DiT scaling behavior on several generative modeling metrics.** *Left:* We plot model performance as a function of total training compute for FID, sFID, Inception Score, Precision and Recall. *Right:* We plot model performance at 400K training steps for all 12 DiT variants against transformer Gflops, finding strong correlations across metrics. All values were computed using the ft-MSE VAE decoder.

Figure 13. **Training loss curves for all DiT models.** We plot the loss over training for all DiT models (the sum of the noise prediction mean-squared error and $\mathcal{D}_{KL}$). We also highlight early training behavior. Note that scaled-up DiT models exhibit lower training losses.

Figure 14. **Uncurated** 512×512 **DiT-XL/2 samples.**
Classifier-free guidance scale = 4.0
Class label = “arctic wolf” (270)

Figure 15. **Uncurated** 512×512 **DiT-XL/2 samples.**
Classifier-free guidance scale = 4.0
Class label = “volcano” (980)

Figure 16. **Uncurated** 512×512 **DiT-XL/2 samples.**
Classifier-free guidance scale = 4.0
Class label = “husky” (250)

Figure 17. **Uncurated** 512×512 **DiT-XL/2 samples.**
Classifier-free guidance scale = 4.0
Class label = “sulphur-crested cockatoo” (89)

Figure 18. **Uncurated** 512×512 **DiT-XL/2 samples.**
Classifier-free guidance scale = 4.0
Class label = "cliff drop-off" (972)

Figure 19. **Uncurated** 512×512 **DiT-XL/2 samples.**
Classifier-free guidance scale = 4.0
Class label = "balloon" (417)

Figure 20. **Uncurated** 512×512 **DiT-XL/2 samples.**
Classifier-free guidance scale = 4.0
Class label = “lion” (291)

Figure 21. **Uncurated** 512×512 **DiT-XL/2 samples.**
Classifier-free guidance scale = 4.0
Class label = “otter” (360)

Figure 22. **Uncurated** 512×512 **DiT-XL/2 samples.**
Classifier-free guidance scale = 2.0
Class label = "red panda" (387)

Figure 23. **Uncurated** 512×512 **DiT-XL/2 samples.**
Classifier-free guidance scale = 2.0
Class label = "panda" (388)

Figure 24. **Uncurated** 512×512 **DiT-XL/2 samples.**
Classifier-free guidance scale = 1.5
Class label = "coral reef" (973)

Figure 25. **Uncurated** 512×512 **DiT-XL/2 samples.**
Classifier-free guidance scale = 1.5
Class label = "macaw" (88)

Figure 26. **Uncurated** 256×256 **DiT-XL/2 samples.**
Classifier-free guidance scale = 4.0
Class label = “macaw” (88)

Figure 27. **Uncurated** 256×256 **DiT-XL/2 samples.**
Classifier-free guidance scale = 4.0
Class label = “dog sled” (537)

Figure 28. **Uncurated** 256×256 **DiT-XL/2 samples.**
Classifier-free guidance scale = 4.0
Class label = “arctic fox” (279)

Figure 29. **Uncurated** 256×256 **DiT-XL/2 samples.**
Classifier-free guidance scale = 4.0
Class label = “loggerhead sea turtle” (33)

Figure 30. **Uncurated** 256×256 **DiT-XL/2 samples.**
Classifier-free guidance scale = 2.0
Class label = "golden retriever" (207)

Figure 31. **Uncurated** 256×256 **DiT-XL/2 samples.**
Classifier-free guidance scale = 2.0
Class label = "lake shore" (975)

Figure 32. **Uncurated** 256×256 **DiT-XL/2 samples.**
Classifier-free guidance scale = 1.5
Class label = "space shuttle" (812)

Figure 33. **Uncurated** 256×256 **DiT-XL/2 samples.**
Classifier-free guidance scale = 1.5
Class label = "ice cream" (928)