

# DYNAMIC DIFFUSION TRANSFORMER

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

Diffusion Transformer (DiT), an emerging diffusion model for image generation, has demonstrated superior performance but suffers from substantial computational costs. Our investigations reveal that these costs stem from the *static* inference paradigm, which inevitably introduces redundant computation in certain *diffusion timesteps* and *spatial regions*. To address this inefficiency, we propose **Dynamic Diffusion Transformer (DyDiT)**, an architecture that *dynamically* adjusts its computation along both *timestep* and *spatial* dimensions during generation. Specifically, we introduce a *Timestep-wise Dynamic Width* (TDW) approach that adapts model width conditioned on the generation timesteps. In addition, we design a *Spatial-wise Dynamic Token* (SDT) strategy to avoid redundant computation at unnecessary spatial locations. Extensive experiments on various datasets and different-sized models verify the superiority of DyDiT. Notably, with <3% additional fine-tuning iterations, our method reduces the FLOPs of DiT-XL by 51%, accelerates generation by 1.73 $\times$ , and achieves a competitive FID score of 2.07 on ImageNet.

## 1 INTRODUCTION

Diffusion models (Ho et al., 2020; Dhariwal & Nichol, 2021; Rombach et al., 2022; Blattmann et al., 2023) have demonstrated significant superiority in visual generation tasks. Recently, the remarkable scalability of Transformers (Vaswani et al., 2017; Dosovitskiy et al., 2020) has led to the growing prominence of Diffusion Transformer (DiT) (Peebles & Xie, 2023). DiT has shown strong potential across a variety of generation tasks (Chen et al., 2023; Ma et al., 2024; Chen et al., 2024) and is considered a foundational component in the development of Sora (Brooks et al., 2024), a pioneering model for video generation. Like Transformers in other vision and language domains (Dosovitskiy et al., 2020; Brown et al., 2020), DiT faces significant efficiency challenges during generation.

Existing approaches to improving DiT’s efficiency include efficient diffusion samplers (Song et al., 2020a; 2023; Salimans & Ho, 2022; Meng et al., 2023; Luo et al., 2023) and global acceleration techniques (Ma et al., 2023; Pan et al., 2024). In addition, reducing computational redundancy within the DiT architecture using model compression techniques, such as structural pruning (Fang et al., 2024; Molchanov et al., 2016; He et al., 2017), also shows significant promise.

However, pruning methods typically retain a *static* architecture across both the *timestep* and *spatial* dimensions throughout the diffusion process. As shown in Figure 1(c), both the original DiT and the pruned DiT employ a fixed model width across all diffusion timesteps and allocate the same computational cost to every image patch. This static inference paradigm overlooks the varying complexities associated with different timesteps and spatial regions, leading to significant computational inefficiency. To explore this redundancy in more detail, we analyze the training process of DiT, during which it is optimized for a noise prediction task. Our analysis yields two key insights:

*a) Timestep perspective:* We plot the loss value differences between a pre-trained small model (DiT-S) and a larger model (DiT-XL) in Figure 1(a). The results show that the loss differences diminish substantially for  $t > \hat{t}$ , and even approach negligible levels as  $t$  nears the prior distribution ( $t \rightarrow T$ ). This indicates that the prediction task becomes *progressively easier at later timesteps* and could be managed effectively even by a *smaller model*. However, DiT applies the same architecture across all timesteps, leading to *excessive computational costs at timesteps where the task complexity is low*.

*b) Spatial perspective:* We visualize the loss maps in Figure 1(b) and observe a noticeable imbalance in loss values across different spatial regions of the image. Loss values are higher in patches corresponding to the main object, while patches representing background regions exhibit relatively lower

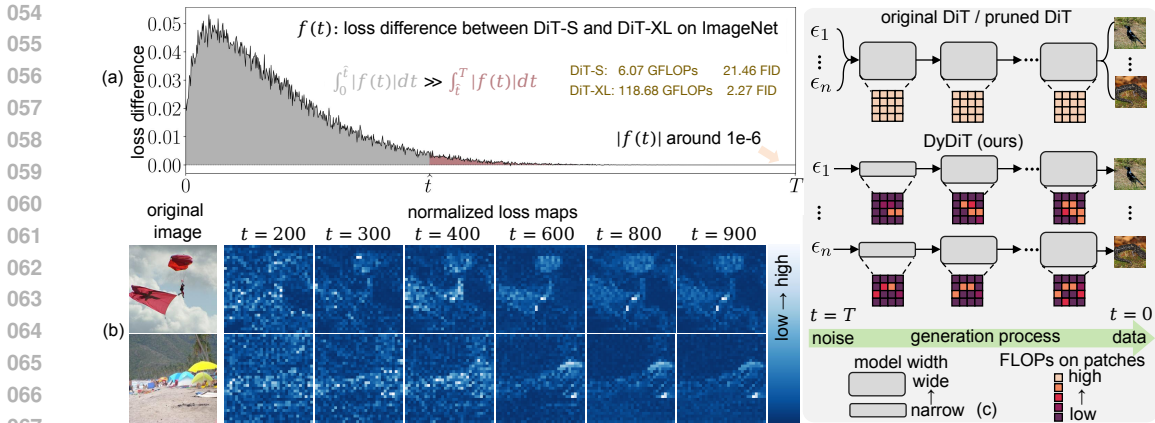


Figure 1: (a) The loss difference between DiT-S and DiT-XL across all diffusion timesteps ( $T = 1000$ ). The difference is slight at most timesteps. (b) Loss maps (normalized to the range  $[0, 1]$ ) at different timesteps, show that the noise in different patches has varying levels of difficulty to predict. (c) Difference of the inference paradigm between the static DiT and the proposed DyDiT.

loss. This suggests that the difficulty of noise prediction varies across spatial regions. Consequently, *uniform computational treatment of all patches introduces redundancy and is likely suboptimal*.

Based on the above insights, a promising approach to improve DiT’s computational efficiency is *dynamic computation*. To this end, we propose **Dynamic Diffusion Transformer (DyDiT)**, which adaptively allocates computational resources during the generation process, as illustrated in Figure 1(c). Specifically, from the timestep perspective, we introduce a *Timestep-wise Dynamic Width (TDW)* mechanism, where the model learns to adjust the width of the attention and MLP blocks based on the current *timestep*. From a spatial perspective, we develop a *Spatial-wise Dynamic Token (SDT)* strategy, which identifies image patches where noise prediction is relatively “easy”, allowing them to bypass computationally intensive blocks, thus reducing unnecessary computation.

Notably, both TDW and SDT are plug-and-play modules that can be easily implemented on DiT to build DyDiT. Moreover, our method contributes to significant speedup due to the hardware-friendly design: 1) the model architecture at each timestep can be pre-determined offline, eliminating additional overhead for width adjustments and enabling efficient batch processing (Section 3.2); and 2) the token gathering and scattering operations incur minimal overhead and are straightforward to implement (Section 3.3). Such hardware efficiency distinguishes our approach from traditional dynamic networks (Herrmann et al., 2020; Meng et al., 2022; Han et al., 2023b), which adapt their inference graphs for each sample and struggle to improve practical efficiency in batched inference.

We conduct extensive experiments across multiple datasets and model scales to validate the effectiveness of the proposed method. For example, compared to the static counterpart DiT-XL, our DyDiT-XL reduces FLOPs by 51% and accelerates the generation by 1.73 times, with less than 3% fine-tuning iterations, while maintaining a competitive FID score of 2.07 on ImageNet ( $256 \times 256$ ) (Deng et al., 2009). Our method shows potential for further efficiency gains when combined with efficient samplers, such as DDIM (Song et al., 2020a) and DPM Solver++ (Lu et al., 2022), or global acceleration techniques like DeepCache (Ma et al., 2023). We anticipate that DyDiT will inspire future research in the development of more efficient diffusion Transformers.

## 2 RELATED WORKS

**Efficient Diffusion Models.** Although diffusion models (Ho et al., 2020; Rombach et al., 2022) have achieved remarkable performance in generation tasks, their generation speed has always hindered their further applications primarily due to long sampling steps and high computational costs. Existing attempts to make diffusion models efficient can be roughly categorized into sampler-based methods, model-based methods, and global acceleration methods. The sampler-based methods (Song et al., 2020a; 2023; Salimans & Ho, 2022; Meng et al., 2023; Luo et al., 2023) aim to reduce the sampling steps. Model-based approaches (Fang et al., 2024; So et al., 2024; Shang et al., 2023; Yang et al., 2023) attempt to compress the size of diffusion models via pruning (Fang et al., 2024; Shang et al., 2023) or

quantization (Li et al., 2023; Shang et al., 2023). Global acceleration methods like Deepcache (Ma et al., 2023) tend to reuse or share some features across different timesteps.

Our DyDiT is mostly relates to the model-based approaches and orthogonal to the other two lines of work. However, unlike the pruning methods yielding *static architectures*, DyDiT performs *dynamic computation* for different diffusion timesteps and image tokens.

**Dynamic Neural Networks.** Compared to static models, dynamic neural networks (Han et al., 2021) can adapt their computational graph based on inputs, enabling superior trade-off between performance and efficiency. They generally realize dynamic architectures by varying the network depth (Teerapittayanon et al., 2016; Bolukbasi et al., 2017; Yang et al., 2020; Han et al., 2022; 2023a) or width (Herrmann et al., 2020; Li et al., 2021; Han et al., 2023b) during inference. Some works explore the spatial redundancy in image *recognition* (Wang et al., 2021; Song et al., 2021; Rao et al., 2021; Liang et al., 2022; Meng et al., 2022). Despite their promising theoretical efficiency, existing dynamic networks usually struggle in achieving *practical efficiency* during batched inference (Han et al., 2023b) due to the per-sample inference graph. Moreover, the potential of dynamic architectures in diffusion models, where a *timestep* dimension is introduced, remains unexplored.

This work extends the research of dynamic networks to the image *generation* field. More importantly, our TDW adjusts the network structure only conditioned on the *timesteps*, avoiding the sample-conditioned tensor shapes in batched inference. Together with the efficient token gathering and scattering mechanism of SDT, DyDiT shows preferable realistic efficiency.

### 3 DYNAMIC DIFFUSION TRANSFORMER

We first provide an overview of diffusion models and DiT (Peebles & Xie, 2023) in Section 3.1. DyDiT’s timestep-wise dynamic width (TDW) and spatial-wise dynamic token (SDT) approaches are then introduced in Sections 3.2 and 3.3. Finally, Section 3.4 details the training process of DyDiT.

#### 3.1 PRELIMINARY

**Diffusion Models** (Ho et al., 2020; Song et al., 2020b; Nichol & Dhariwal, 2021; Rombach et al., 2022) generate images from random noise through a series of diffusion steps. These models typically consist of a forward diffusion process and a reverse denoising process. In the forward process, given an image  $\mathbf{x}_0 \sim q(\mathbf{x})$  sampled from the data distribution, Gaussian noise  $\epsilon \sim \mathcal{N}(0, I)$  is progressively added over  $T$  steps. This process is defined as  $q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I})$ , where  $t$  and  $\beta_t$  denote the timestep and noise schedule, respectively. In the reverse process, the model removes the noise and reconstructs  $\mathbf{x}_0$  from  $\mathbf{x}_T \sim \mathcal{N}(0, I)$  using  $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$ , where  $\mu_\theta(\mathbf{x}_t, t)$  and  $\Sigma_\theta(\mathbf{x}_t, t)$  represent the mean and variance of the Gaussian distribution.

**Diffusion Transformer (DiT)** (Peebles & Xie, 2023) exhibits the scalability and promising performance of Transformers (Brooks et al., 2024). Similar to ViT (Dosovitskiy et al., 2020), DiT consists of layers composed of a multi-head self-attention (MHSA) block and a multi-layer perceptron (MLP) block, described as  $\mathbf{X} \leftarrow \mathbf{X} + \alpha \text{MHSA}(\gamma \mathbf{X} + \beta)$ ,  $\mathbf{X} \leftarrow \mathbf{X} + \alpha' \text{MLP}(\gamma' \mathbf{X} + \beta')$ , where  $\mathbf{X} \in \mathbb{R}^{N \times C}$  denotes image tokens. Here,  $N$  is the number of tokens, and  $C$  is the channel dimension. The parameters  $\{\alpha, \gamma, \beta, \alpha', \gamma', \beta'\}$  are produced by an adaptive layer norm (adaLN) block (Perez et al., 2018), which takes the class condition embedding  $\mathbf{E}_{cls}$  and timestep embedding  $\mathbf{E}_t$  as inputs.

#### 3.2 TIMESTEP-WISE DYNAMIC WIDTH

As aforementioned, DiT spends equal computation for different timesteps, although not all steps share the same generation difficulty (Figure 1(a)). Therefore, the *static* computation paradigm introduces significant redundancy in those “easy” timesteps. Inspired by structural pruning methods (He et al., 2017; Hou et al., 2020; Fang et al., 2024), we propose a *timestep-wise dynamic width (TDW)* mechanism, which adjusts the width of MHSA and MLP blocks in different timesteps. Note that TDW is *not a pruning method* that permanently removes certain model components, but rather retains the full capacity of DiT and *dynamically* activates different heads/channel groups at each timestep.

**Heads and channel groups.** Given input  $\mathbf{X} \in \mathbb{R}^{N \times C}$ , an MHSA block employs three linear layers with weights  $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{C \times (H \times C_H)}$  to project it into Q, K, and V features, respectively. Here,  $H$  denotes the head number and  $C = H \times C_H$  in DiT. An output projection is performed using another

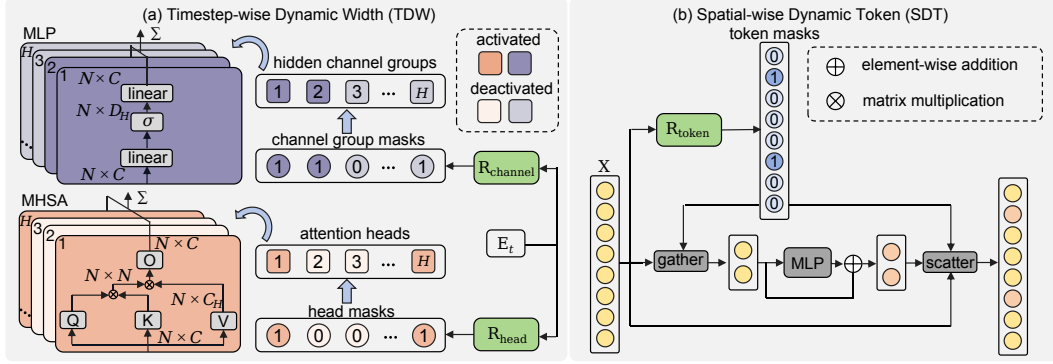


Figure 2: **Overview of the proposed dynamic diffusion transformer (DyDiT).** It reduces the computational redundancy in DiT (Peebles & Xie, 2023) from both timestep and spatial dimensions.

linear layer with  $\mathbf{W}_O \in \mathbb{R}^{(H \times C_H) \times C}$ . The operation of the conventional MHA can be expressed as:

$$\text{MHA}(\mathbf{X}) = \sum_{h=1}^H \mathbf{X}_{\text{attn}}^h \mathbf{W}_O^{h,:} = \sum_{h=1}^H (\text{Softmax}((\mathbf{X}\mathbf{W}_Q^{:,h,:})(\mathbf{X}\mathbf{W}_K^{:,h,:})^\top) \mathbf{X}\mathbf{W}_V^{:,h,:}) \mathbf{W}_O^{h,:}. \quad (1)$$

An MLP block contains two linear layers with weights  $\mathbf{W}_1 \in \mathbb{R}^{C \times D}$  and  $\mathbf{W}_2 \in \mathbb{R}^{D \times C}$ , where  $D$  represents the hidden channels, set as  $4C$  by default in DiT. To dynamically control the MLP width, we divide  $D$  hidden channels into  $H$  groups, reformulating the weights into  $\mathbf{W}_1 \in \mathbb{R}^{C \times (H \times D_H)}$  and  $\mathbf{W}_2 \in \mathbb{R}^{(H \times D_H) \times C}$ , where  $D_H = D/H$ . Hence, the operation in MLP can be formulated as:

$$\text{MLP}(\mathbf{X}) = \sum_{h=1}^H \sigma(\mathbf{X}_{\text{hidden}}^h) \mathbf{W}_2^{h,:} = \sum_{h=1}^H \sigma(\mathbf{X}\mathbf{W}_1^{:,h,:}) \mathbf{W}_2^{h,:}, \quad (2)$$

where  $\sigma$  denotes the activation layer.

**Dynamic width control based on timestep.** To dynamically activate the heads and channel groups at each diffusion timestep, in each block, we feed the timestep embedding  $\mathbf{E}_t \in \mathbb{R}^C$  into routers  $R_{\text{head}}$  and  $R_{\text{channel}}$  (Figure 2(a)). Each router comprises a linear layer followed by the Sigmoid function, producing the probability of each head and channel group to be activated:

$$\mathbf{S}_{\text{head}} = R_{\text{head}}(\mathbf{E}_t) \in [0, 1]^H, \quad \mathbf{S}_{\text{channel}} = R_{\text{channel}}(\mathbf{E}_t) \in [0, 1]^H. \quad (3)$$

A threshold of 0.5 is then used to convert the continuous-valued  $\mathbf{S}_{\text{head}}$  and  $\mathbf{S}_{\text{channel}}$  into binary masks  $\mathbf{M}_{\text{head}} \in \{0, 1\}^H$  and  $\mathbf{M}_{\text{channel}} \in \{0, 1\}^H$ , indicating the activation decisions for attention heads and channel groups. The  $h$ -th head (group) is activated only when  $\mathbf{M}_{\text{head}}^h = 1$  ( $\mathbf{M}_{\text{channel}}^h = 1$ ). Benefiting from the grouping operation, routers introduce negligible parameters and computation.

**Inference.** After obtaining the discrete decisions  $\mathbf{M}_{\text{head}}$  and  $\mathbf{M}_{\text{channel}}$ , each DyDiT block only computes the activated heads and channel groups during generation:

$$\begin{aligned} \text{MHA}(\mathbf{X}) &= \sum_{h:\mathbf{M}_{\text{head}}^h=1} \mathbf{X}_{\text{attn}}^h \mathbf{W}_O^{h,:}, \\ \text{MLP}(\mathbf{X}) &= \sum_{h:\mathbf{M}_{\text{channel}}^h=1} \sigma(\mathbf{X}_{\text{hidden}}^h) \mathbf{W}_2^{h,:}. \end{aligned} \quad (4)$$

Let  $\tilde{H}_{\text{head}} = \sum_h \mathbf{M}_{\text{head}}^h$  and  $\tilde{H}_{\text{channel}} = \sum_h \mathbf{M}_{\text{channel}}^h$  denote the number of activated heads/groups. TWD reduces the MHA computation from  $\mathcal{O}(H \times (4NCC_H + 2N^2C_H))$  to  $\mathcal{O}(\tilde{H}_{\text{head}} \times (4NCC_H + 2N^2C_H))$  and MLP blocks from  $\mathcal{O}(H \times 2NCD_H)$  to  $\mathcal{O}(\tilde{H}_{\text{channel}} \times 2NCD_H)$ . It is worth noting that as the activation choices depend solely on the timestep  $\mathbf{E}_t$ , we can pre-compute the masks offline once the training is completed, and *pre-define* the activated network architecture before deployment. This avoids the sample-dependent inference graph in traditional dynamic architectures (Meng et al., 2022; Han et al., 2023b) and facilitates the realistic speedup in batched inference.

### 216 3.3 SPATIAL-WISE DYNAMIC TOKEN

217 In addition to the timestep dimension, the redundancy widely exists in the spatial dimension due to  
 218 the varying complexity of different patches (Figure 1(b)). To this end, we propose a spatial-wise  
 219 dynamic token (SDT) method to reduce computation for the patches where noise estimation is “easy”.

220 **Bypassing the MLP block.** As shown in Figure 2 (b), SDT adaptively identifies the tokens associated  
 221 with image regions that present lower noise prediction difficulty. These tokens are then allowed to  
 222 bypass the computationally intensive MLP blocks. Theoretically, this block-bypassing operation  
 223 can be applied to both MHSA and MLP. However, we find MHSA crucial for establishing token  
 224 interactions, which is essential for the generation quality. More critically, varying token numbers  
 225 across images in MHSA could result in incomplete tensor shapes in a batch, reducing the overall  
 226 throughput in generation. Therefore, SDT is applied only in MLP blocks in each layer.

227 Concretely, before each MLP block, we feed the input  $\mathbf{X} \in \mathbb{R}^{N \times C}$  into a token router  $R_{\text{token}}$ , which  
 228 predicts the probability  $\mathbf{S}_{\text{token}} \in \mathbb{R}^N$  of each token to be processed. This can be formulated as:

$$230 \mathbf{S}_{\text{token}} = R_{\text{token}}(\mathbf{X}) \in [0, 1]^N. \quad (5)$$

231 We then convert it into a binary mask  $\mathbf{M}_{\text{token}}$  using a threshold of 0.5. Each element  $\mathbf{M}_{\text{token}}^i \in \{0, 1\}$   
 232 in the mask indicates whether the  $i$ -th token should be processed by the block (if  $\mathbf{M}_{\text{token}}^i = 1$ ) or  
 233 directly bypassed (if  $\mathbf{M}_{\text{token}}^i = 0$ ). The router parameters are not shared across different layers.

234 **Inference.** During inference (Figure 2(b)), we gather the tokens based on the mask  $\mathbf{M}_{\text{token}}$  and feed  
 235 them to the MLP, thereby avoiding unnecessary computational costs for other tokens. Then, we adopt  
 236 a scatter operation to reposition the processed tokens. This further reduces the computational cost  
 237 of the MLP block from  $\mathcal{O}(\tilde{H}_{\text{channel}}N \times 2CD_H)$  to  $\mathcal{O}(\tilde{H}_{\text{channel}}\tilde{N} \times 2CD_H)$ , where  $\tilde{N} = \sum_i \mathbf{M}_{\text{token}}^i$   
 238 denotes the actual number of tokens to be processed. Since there is no token interaction within the  
 239 MLP, the SDT operation supports batched inference, improving the practical generation efficiency.

### 241 3.4 FLOPS-AWARE END-TO-END TRAINING

242 In the following, we first present the details of end-to-end training, followed by the loss design for  
 243 controlling the computational complexity of DyDiT and techniques to stabilize fine-tuning.

244 **End-to-end training.** During training, in TWD, we multiply  $\mathbf{M}_{\text{head}}$  and  $\mathbf{M}_{\text{channel}}$  with their corre-  
 245 sponding features ( $\mathbf{X}_{\text{attn}}$  and  $\mathbf{X}_{\text{hidden}}$ ) to zero out the deactivated heads and channel groups, respec-  
 246 tively. Similarly, in SDT, we multiply  $\mathbf{M}_{\text{token}}$  with  $\text{MLP}(\mathbf{X})$  to deactivate the tokens that should not  
 247 be processed by MLP. Straight-through-estimator (Bengio et al., 2013) and Gumbel-Sigmoid (Meng  
 248 et al., 2022) are employed to enable the end-to-end training of routers.

249 **Training with FLOPs-constrained loss.** We design a FLOPs-constrained loss to control the compu-  
 250 tational cost during the generation process. We find it impractical to obtain the entire computation  
 251 graph during  $T$  timesteps since the total timestep  $T$  is large *e.g.*  $T = 1000$ . Fortunately, the timesteps  
 252 in a batch are sampled from  $t \sim \text{Uniform}(0, T)$  during training, which approximately covers the  
 253 entire computation graph. Let  $B$  denote the batch size, with  $t_b$  as the timestep for the  $b$ -th sample, we  
 254 compute the total FLOPs at the sampled timestep,  $F_{\text{dynamic}}^{t_b}$ , using masks  $\{\mathbf{M}_{\text{head}}^{t_b}, \mathbf{M}_{\text{channel}}^{t_b}, \mathbf{M}_{\text{token}}^{t_b}\}$   
 255 from each transformer layer, as detailed in Section 3.2 and Section 3.3. Let  $F_{\text{static}}$  denote the total  
 256 FLOPs of MHSA and MLP blocks in the static DiT. We formulate the FLOPs-constrained loss as:

$$258 \mathcal{L}_{\text{FLOPs}} = \left( \frac{1}{B} \sum_{t_b: b \in [1, B]} \frac{F_{\text{dynamic}}^{t_b}}{F_{\text{static}}} - \lambda \right)^2, \quad (6)$$

260 where  $\lambda$  is a hyperparameter representing the target FLOPs ratio, and  $t_b$  is uniformly sampled from  
 261 the interval  $[0, T]$ . The overall training objective combines this FLOPs-constrained loss with the  
 262 original DiT training loss, expressed as  $\mathcal{L} = \mathcal{L}_{\text{DiT}} + \mathcal{L}_{\text{FLOPs}}$ .

263 **Fine-tuning stabilization.** In practice, we find directly finetuning DyDiT with  $\mathcal{L}$  might occasionally  
 264 lead to unstable training. To address this, we employ two stabilization techniques. First, for a warm-  
 265 up phase we maintain a complete DiT model supervised by the same diffusion target, introducing an  
 266 additional item,  $\mathcal{L}_{\text{DiT}}^{\text{complete}}$  along with  $\mathcal{L}$ . After this phase, we remove this item and continue training  
 267 solely with  $\mathcal{L}$ . Additionally, prior to fine-tuning, we rank the heads and hidden channels in MHSA  
 268 and MLP blocks based on a magnitude criterion (He et al., 2017). We consistently select the most  
 269 important head and channel group in TDW. This ensures that at least one head and channel group is  
 activated in each MHSA and MLP block across all timesteps, thereby alleviating the instability.

Table 1: **Comparison with state-of-the-art diffusion models on ImageNet  $256 \times 256$ .** DyDiT-XL achieves competitive performance while significantly reducing the computational cost.

Model	Params. (M) ↓	FLOPs (G) ↓	FID ↓	sFID ↓	IS ↑	Precision ↑	Recall ↑
<i>Static</i>							
ADM	608	1120	4.59	5.25	186.87	0.82	0.52
LDM-4	400	104	3.95	-	178.22	0.81	0.55
U-ViT-L/2	<b>287</b>	<u>77</u>	3.52	-	-	-	-
U-ViT-H/2	501	113	2.29	-	247.67	<b>0.87</b>	0.48
DiffuSSM-XL	673	280	2.28	<b>4.49</b>	269.13	<u>0.86</u>	0.57
DiM-L	<u>380</u>	94	2.64	-	-	-	-
DiM-H	860	210	2.21	-	-	-	-
DiT-L	468	81	5.02	-	167.20	0.75	0.57
DiT-XL	675	118	2.27	4.60	277.00	0.83	0.57
<i>Dynamic</i>							
DyDiT-XL $_{\lambda=0.7}$	678	84.33	<u>2.12</u>	4.61	<b>284.31</b>	0.81	<u>0.60</u>
DyDiT-XL $_{\lambda=0.5}$	678	<b>57.88</b>	<b>2.07</b>	<u>4.56</u>	248.03	0.80	<b>0.61</b>

## 4 EXPERIMENTS

**Implementation details.** Our DyDiT can be built easily by fine-tuning on pre-trained DiT weights. We experiment on three different-sized DiT models denoted as DiT-S/B/XL. For DiT-XL, we directly adopt the checkpoint from the official DiT repository (Peebles & Xie, 2023), while for DiT-S and DiT-B, we use pre-trained models provided in Pan et al. (2024). All experiments are conducted on a server with  $8 \times$  NVIDIA A800 80G GPUs. More details of model configurations and training setup can be found in Appendix A.1 and A.2, respectively. Following DiT (Peebles & Xie, 2023), the strength of classifier-free guidance (Ho & Salimans, 2022) is set to 1.5 and 4.0 for evaluation and visualization, respectively. Unless otherwise specified, 250 DDPM (Ho et al., 2020) sampling steps are used. All speed tests are performed on an NVIDIA V100 32G GPU.

**Datasets.** Following the protocol in DiT (Peebles & Xie, 2023), we mainly conduct experiments on ImageNet (Deng et al., 2009) at a resolution of  $256 \times 256$ . To comprehensively evaluate our method, we also assess performance and efficiency on four fine-grained datasets used by Xie et al. (2023): Food (Bossard et al., 2014), Artbench (Liao et al., 2022), Cars (Gebru et al., 2017) and Birds (Wah et al., 2011). We conduct experiments in both in-domain fine-tuning and cross-domain transfer learning manners on these dataset. Images of these datasets are also resized into  $256 \times 256$  resolution.

**Metrics.** Following prior works (Peebles & Xie, 2023; Teng et al., 2024), we sample 50,000 images to measure the Fréchet Inception Distance (FID) (Heusel et al., 2017) score with the ADM’s TensorFlow evaluation suite (Dhariwal & Nichol, 2021). Inception Score (IS) (Salimans et al., 2016), sFID (Nash et al., 2021), and Precision-Recall (Kynkäänniemi et al., 2019) are also reported for complementary. **Bold font** and underline denote the best and the second-best performance, respectively.

### 4.1 COMPARISON WITH STATE-OF-THE-ART DIFFUSION MODELS

In Table 1, we compare our method with other representative diffusion models, including ADM (Dhariwal & Nichol, 2021), LDM (Rombach et al., 2022), U-ViT (Bao et al., 2023), DiffuSSM (Yan et al., 2024), DiM (Teng et al., 2024), and DiT (Peebles & Xie, 2023), on  $256 \times 256$  ImageNet generation. All methods except ours adopt a static architecture. DyDiT-XL is fine-tuned with fewer than 3% additional iterations based on DiT to adapt the dynamic architecture, as detailed in Appendix A.1.

Notably, our model DyDiT $_{\lambda=0.5}$  achieves a 2.07 FID score with less than 50% FLOPs of its counterpart, DiT-XL, and outperforms other models obviously. This verify that our method can effectively remove the redundant computation in DiT and maintain the generation performance. It accelerates the generation speed by  $1.73 \times$ , which is detailed in Appendix B.1. Increasing the target FLOPs ratio  $\lambda$  from 0.5 to 0.7 enables DyDiT $_{\lambda=0.7}$  to achieve competitive performance with DiT-XL across most metrics and obtains the best IS score. This improvement is likely due to DyDiT’s dynamic architecture, which offers greater flexibility compared to static architectures, allowing the model to address each timestep and image patch specifically during the generation process. With around 80G FLOPs, our DyDiT $_{\lambda=0.7}$  method significantly outperforms U-ViT-L/2 and DiT-L, further validating the advantages of our dynamic generation paradigm.

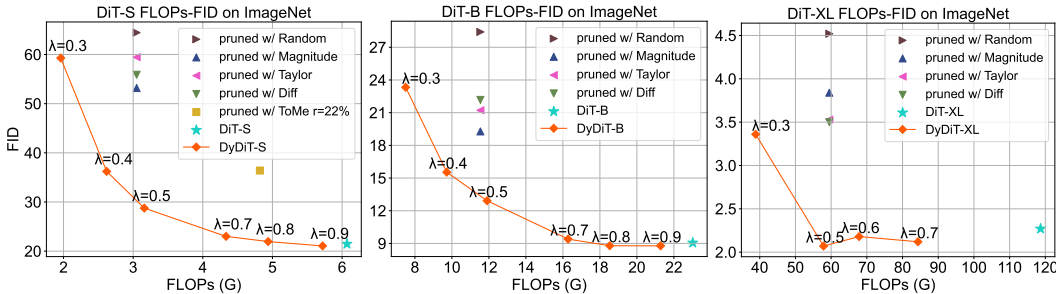


Figure 3: **FLOPs-FID trade-off for S, B, and XL size models on ImageNet.** For clarity, we omit the results of applying ToMe to DiT-B and DiT-XL, as it does not surpass the random pruning.

## 4.2 COMPARISON WITH PRUNING METHODS

**Benchmarks.** The proposed timestep-wise dynamic width and spatial-wise dynamic token improve efficiency from the model *architecture* and *token* redundancy perspective, respectively. To evaluate the superiority of our approach, we compare our methods against representative *static* structure and token pruning techniques. More details of this experiment can be found in Appendix A.3.

*Pruning-based methods.* We include Diff pruning Fang et al. (2024) in the comparison, which is a Taylor-based (Molchanov et al., 2016) pruning method specifically optimized for the diffusion process and has demonstrated superiority on diffusion models with U-Net (Ronneberger et al., 2015) architecture (Fang et al., 2024). Following Fang et al. (2024), we also include Random pruning, Magnitude pruning (He et al., 2017), and Taylor pruning (Molchanov et al., 2016) in the comparison. We adopt these four pruning approaches to distinguish important heads and channels in DiT from less significant ones, which can be removed to reduce the model width.

*Token merging.* We also compare our methods with a training-free token pruning technique, ToMe (Bolya et al., 2022), which progressively prunes tokens in each vision transformer (Dosovitskiy et al., 2020) layer through adaptive token merging. Its enhanced version (Bolya & Hoffman, 2023) can also accelerate diffusion models based on U-Net architectures *e.g.* Stable Diffusion Rombach et al. (2022). We directly apply the enhanced version in each layer of DiT.

**Results.** We present the FLOPs-FID curves for S, B, and XL size models in Figure 3. Across different sizes, DyDiT significantly outperforms all pruning methods with similar or even lower FLOPs, highlighting the superiority of dynamic architecture over static pruning in diffusion transformers.

Interestingly, Magnitude pruning shows slightly better performance among structural pruning techniques on DiT-S and DiT-B, while Diff pruning and Taylor pruning perform better on DiT-XL. This indicates that different-sized DiT prefer distinct pruning criteria. Although ToMe (Bolya & Hoffman, 2023) successfully accelerates U-Net models with acceptable performance loss, its application to DiT results in performance degradation, as also observed in Moon et al. (2023). We conjecture that the errors introduced by token merging become irrecoverable in DiT due to the absence of convolutional layers and long-range skip connections present in U-Net architectures.

**Scaling up ability.** We can observe from Figure 3 that the performance gap between DyDiT and DiT diminishes as model size increases. Specifically, DyDiT-S achieves a comparable FID to the original DiT only at  $\lambda = 0.9$ , while DyDiT-B achieves this with a lower FLOPs ratio, *e.g.*,  $\lambda = 0.7$ . When scaled to XL, DyDiT-XL attains a slightly better FID even at  $\lambda = 0.5$ . This is due to increased computation redundancy with larger models, allowing our method to reduce redundancy without compromising FID. These results validate the scalability of our approach, which is crucial in the era of large models, encouraging further exploration of larger models in the future.

## 4.3 RESULTS ON FINE-GRAINED DATASETS

**Quantitative results.** We further compare our method with structural pruning and token pruning approaches on fine-grained datasets under the in-domain fine-tuning setting, where the DiT is initially pre-trained on the corresponding dataset and subsequently fine-tuned on the same dataset for pruning or dynamic adaptation. Detailed experiment settings are presented in Appendix A.4. Results are summarized in Table 2. With the pre-defined FLOPs ratio  $\lambda = 0.5$ , our method significantly reduces computational cost and enhances generation speed while maintaining performance levels comparable to the original DiT. To ensure fair comparisons, we set width pruning ratios to 50% for pruning

Table 2: **Results on fine-grained datasets.** The model marked with † corresponds to fine-tuning directly on the target dataset. See the main texts for details.

Model	s/image ↓	FLOPs (G) ↓	Food	Artbench	FID ↓ Cars	Birds	#Average
DiT-S	0.65	6.07	<u>14.56</u>	<b>17.54</b>	<b>9.30</b>	<b>7.69</b>	<b>12.27</b>
pruned w/ random	0.38	3.05	45.66	76.75	60.26	48.60	57.81
pruned w/ magnitude	0.38	3.05	41.93	42.04	31.49	26.45	35.44
pruned w/ taylor	0.38	3.05	47.26	74.21	27.19	22.33	42.74
pruned w/ diff	0.38	3.05	36.93	68.18	26.23	23.05	38.59
pruned w/ ToMe 20%	0.61	4.82	43.87	62.96	32.16	15.20	38.54
DyDiT-S $_{\lambda=0.5}$	0.41	3.16	16.74	21.35	<u>10.01</u>	<u>7.85</u>	13.98
DyDiT-S $_{\lambda=0.5}$ †	0.41	3.17	<b>13.03</b>	<u>19.47</u>	12.15	8.01	<u>13.16</u>



Figure 4: **Qualitative comparison of images generated by the original DiT, DiT pruned with magnitude, and DyDiT.** All models are of “S” size. The FLOPs ratio  $\lambda$  in DyDiT is set to 0.5.

methods, aiming for similar FLOPs. Among structural pruning techniques, Magnitude pruning shows relatively better performance, yet DyDiT consistently outperforms it by a substantial margin. With a 20% merging ratio, ToMe also speeds up generation but sacrifices performance. As mentioned, the lack of convolutional layers and skip connections makes applying ToMe to DiT suboptimal.

**Qualitative visualization.** Figure 4 presents images generated by DyDiT-S on fine-grained datasets, compared to those produced by the original or pruned DiT-S. These qualitative results demonstrate that our method maintains the FID score while producing images of quality comparable to DiT-S.

**Cross-domain transfer learning.** Transferring to downstream datasets is a common practice to leverage pre-trained generations models. In this experiment, we fine-tune a model pre-trained on ImageNet to perform cross-domain adaptation on the target dataset while concurrently learning the dynamic architecture, yielding DyDiT-S $_{\lambda=0.5}$ † in Table 2. More details are presented in Appendix A.5. We can observe that learning the dynamic architecture during the cross domain transfer learning does not hurt the performance, and even leads to slight better average FID score than DyDiT-S $_{\lambda=0.5}$ . This further broadens the application scope of our method.

#### 4.4 ABLATION STUDY

**Main components.** We first conduct experiments to verify the effectiveness of each component in our method. We summarize the results in Table 3. “I” and “II” denote DiT with only the proposed timestep-wise dynamic width (TDW) and spatial-wise dynamic token (SDT), respectively. We can find that “I” performs much better than “II”. This is attributed to the fact that, with the target FLOPs ratio  $\lambda$  set to 0.5, most tokens in “II” have to bypass MLP blocks, leaving only MHSA blocks to process tokens, significantly affecting performance (Dong et al., 2021). “III” represents the default model that combines both TDW and SDT, achieving obviously better performance than “I” and “II”. Given a computational budget, the combination of TDW and SDT allows the model to discover computational redundancy from both the time-step and spatial perspectives.

**Importance of routers in temporal-wise dynamic width.** Routers in TDW adaptively adjust the model width for each block across all timesteps. Replacing the learnable router with a random selection, resulting in “I (random)”, leads to model collapse across all datasets. This is due to the random activation of heads and channel groups, which hinders the model’s ability to generate high-



Table 3: **Ablation Study** on DyDiT-S $_{\lambda=0.5}$ . All models evoke around 3.16 GFLOPs.

Model	TDW	SDT	FID ↓					#Average
			ImageNet	Food	Artbench	Cars	Birds	
I	✓		31.89	<b>15.71</b>	28.19	19.67	9.23	20.93
II		✓	70.06	23.79	52.78	16.90	12.05	35.12
III	✓	✓	<b>28.75</b>	<u>16.74</u>	<b>21.35</b>	<b>10.01</b>	<b>7.85</b>	<b>16.94</b>
I (random)			124.38	111.88	151.99	127.53	164.29	136.01
I (manual)			34.08	23.89	40.02	22.34	20.17	28.10
III (layer-skip)	✓		<u>30.95</u>	17.75	<u>23.15</u>	<u>10.53</u>	<u>9.01</u>	<u>18.29</u>

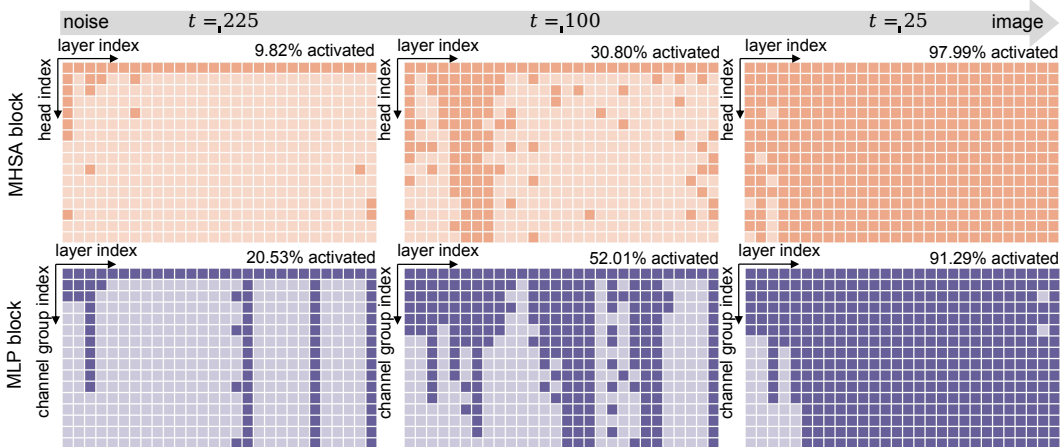


Figure 5: **Visualization of dynamic architecture.** ■ and ■ indicates the deactivated and activated heads in an MHSAs block, while ■ and ■ denotes that the channel group is deactivated or activated in an MLP block, respectively. We conduct 250-step DDPM generation.

quality images. We also experiment a manually-designed strategy, termed “I (manual)”, in which we activate 5/6, 1/2, 1/3, 1/3 of the heads and channels for the intervals  $[0, 1/T]$ ,  $[1/T, 2/T]$ ,  $[2/T, 3/4T]$ , and  $[3/4T, T]$  timesteps, respectively. This results in around 50% average FLOPs reduction. Since this strategy aligns the observation in Figure 1(a) and allocates more computation to timesteps approaching 0, “I (manual)” outperforms “I (random)” obviously. However, it does not surpass “I”, highlighting the importance of learned routers.

**Importance of token-level bypassing in spatial-wise dynamic token.** We also explore an alternative design to conduct token bypassing. Specifically, each MLP block adopts a router to determine whether all tokens of an image should bypass the block. This modification causes SDT to become a layer-skipping approach (Wang et al., 2018). We replace SDT in “III” with this design, resulting in “III (layer-skip)” in Table 3. As outlined in Section 1, varying regions of an image face distinct challenges in noise prediction. A uniform token processing strategy fails to address this heterogeneity effectively. For example, tokens from complex regions might bypass essential blocks, resulting in suboptimal noise prediction. The results presented in Table 3 further confirm that the token-level bypassing in SDT, obviously improves the performance of “III” compared to “III (layer-skip)”.

#### 4.5 VISUALIZATION

**Learned timestep-wise dynamic strategy.** Figure 5 illustrates the activation patterns of heads and channel groups during the 250-step DDPM generation process. Throughout this process, TWD progressively activates more MHSAs heads and MLP channel groups as it transitions from noise to image. As discussed in Section 1, prediction is more straightforward when generation is closer to noise (larger  $t$ ) and becomes increasingly challenging as it approaches the image (smaller  $t$ ). Our visualization corroborates this observation, demonstrating that the model allocates more computational resources to more complex timesteps. Notably, the activation rate of MLP blocks surpasses that of MHSAs blocks at  $t = 255$  and  $t = 100$ . This can be attributed to the token bypass operation in the spatial-wise dynamic token (SDT), which reduces the computational load of MLP blocks, enabling TWD to activate additional channel groups with minimal computational overhead.

**Spatial-wise dynamic token adapts computational cost on each image patch.** We quantify and normalize the computational cost on different image patches during generation, ranging from

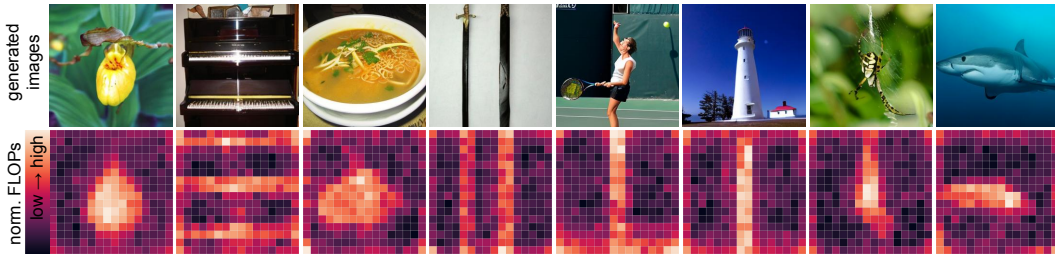


Figure 6: **Computational cost across different image patches.** We quantify the FLOPs cost on image patches over the generation process and normalize them into  $[0, 1]$  for better clarity.

Table 4: **Combination with efficient samplers** (Song et al., 2020a; Lu et al., 2022).

Model	250-DDPM		50-DDIM		20-DPM-solver++		10-DPM-solver++	
	s/image ↓	FID ↓	s/image ↓	FID ↓	s/image ↓	FID ↓	s/image ↓	FID ↓
DiT-XL	10.22	2.27	2.00	2.26	0.84	4.62	0.42	11.66
DyDiT-XL $_{\lambda=0.7}$	7.76	2.12	1.56	2.16	0.62	4.28	0.31	11.10
DyDiT-XL $_{\lambda=0.5}$	5.91	2.07	1.17	2.36	0.46	4.22	0.23	11.31

$[0, 1]$  in Figure 6. These results verify that our SDT effectively learns to adjust computational expenditure based on the complexity of image patches. SDT prioritizes challenging patches containing detailed and colorful main objects. Conversely, it allocates less computation to background regions characterized by uniform and continuous colors. This behavior aligns with our findings in Figure 1(b).

#### 4.6 COMPATIBILITY WITH OTHER EFFICIENT DIFFUSION APPROACHES

**Combination with efficient samplers.** Our DyDiT is a general architecture which can be seamlessly incorporated with efficient samplers such as DDIM (Song et al., 2020a) and DPM-solver++ (Lu et al., 2022). As presented in Table 4, when using the 50-step DDIM, both DiT-XL and DyDiT-XL exhibit significantly faster generation, while our method consistently achieving higher efficiency due to its dynamic computation paradigm. When we further reduce the sampling step to 20 and 10 with DPM-solver++, we observe an FID increasement on all models, while our method still achieves competitive performance compared to the original DiT. These findings highlight the potential of integrating our approach with efficient samplers, suggesting a promising avenue for future research.

**Combination with global acceleration.** DeepCache (Ma et al., 2023) is a train-free technique which globally accelerates generation by caching feature maps at specific timesteps and reusing them in subsequent timesteps. As shown in Table 5, with a cache interval of 2, DyDiT achieves further acceleration with only a marginal performance drop. In contrast, DiT with DeepCache requires a longer interval (e.g. 5) to achieve comparable speed with ours, resulting in an inferior FID score. These results demonstrate the compatibility and effectiveness of our approach in conjunction with DeepCache.

Table 5: **Combination with global acceleration.** “interval” denotes the interval of cached timestep in DeepCache (Ma et al., 2023).

Model	interval	s/image ↓	FID ↓
DiT-XL	0	10.22	2.27
DiT-XL	2	5.02	2.47
DiT-XL	5	2.03	6.73
DyDiT-XL $_{\lambda=0.5}$	0	5.91	2.07
DyDiT-XL $_{\lambda=0.5}$	2	2.99	2.43
DyDiT-XL $_{\lambda=0.5}$	3	2.01	3.37

## 5 DISCUSSION AND CONCLUSION

In this study, we investigate the training process of the Diffusion Transformer (DiT) and identify significant computational redundancy associated with specific diffusion timesteps and image patches. To this end, we propose Dynamic Diffusion Transformer (DyDiT), an architecture that can adaptively adjust the computation allocation across different timesteps and spatial regions. Comprehensive experiments on various datasets and model sizes validate the effectiveness of DyDiT. We anticipate that the proposed method will advance the development of transformer-based diffusion models.

**Limitations and future works.** Similarly to DiT, the proposed DyDiT is currently focusing on image generation. In future works, DyDiT could be further explored to be applied to other tasks, such as video generation (Ma et al., 2024) and controllable generation (Chen et al., 2024).

## REFERENCES

- 540  
541  
542 Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth  
543 words: A vit backbone for diffusion models. In *Proceedings of the IEEE/CVF conference on*  
544 *computer vision and pattern recognition*, pp. 22669–22679, 2023.
- 545  
546 Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through  
547 stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- 548  
549 Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik  
550 Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling  
551 latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.
- 552  
553 Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for  
554 efficient inference. In *ICML*, pp. 527–536. PMLR, 2017.
- 555  
556 Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. In *Proceedings of the*  
557 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4598–4602, 2023.
- 558  
559 Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy  
560 Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022.
- 561  
562 Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101—mining discriminative compo-  
563 nents with random forests. In *ECCV*, pp. 446–461. Springer, 2014.
- 564  
565 Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe  
566 Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video  
567 generation models as world simulators. 2024. URL [https://openai.com/research/  
568 video-generation-models-as-world-simulators](https://openai.com/research/video-generation-models-as-world-simulators).
- 569  
570 Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal,  
571 Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are  
572 few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- 573  
574 Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James  
575 Kwok, Ping Luo, Huchuan Lu, et al. Pixart-alpha: Fast training of diffusion transformer for  
576 photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426*, 2023.
- 577  
578 Junsong Chen, Yue Wu, Simian Luo, Enze Xie, Sayak Paul, Ping Luo, Hang Zhao, and Zhenguo Li.  
579 Pixart- $\delta$ : Fast and controllable image generation with latent consistency models. *arXiv*  
580 *preprint arXiv:2401.05252*, 2024.
- 581  
582 Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale  
583 hierarchical image database. In *CVPR*, pp. 248–255. Ieee, 2009.
- 584  
585 Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances*  
586 *in neural information processing systems*, 34:8780–8794, 2021.
- 587  
588 Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure  
589 attention loses rank doubly exponentially with depth. In *International Conference on Machine*  
590 *Learning*, pp. 2793–2803. PMLR, 2021.
- 591  
592 Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas  
593 Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An  
image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint*  
*arXiv:2010.11929*, 2020.
- Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. *Advances in*  
*neural information processing systems*, 36, 2024.
- Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, and Li Fei-Fei. Fine-grained  
car detection for visual census estimation. In *AAAI*, volume 31, 2017.

- 594 Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural  
595 networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):  
596 7436–7456, 2021.
- 597 Yizeng Han, Yifan Pu, Zihang Lai, Chaofei Wang, Shiji Song, Junfeng Cao, Wenhui Huang, Chao  
598 Deng, and Gao Huang. Learning to weight samples for dynamic early-exiting networks. In *ECCV*,  
599 pp. 362–378. Springer, 2022.
- 600 Yizeng Han, Dongchen Han, Zeyu Liu, Yulin Wang, Xuran Pan, Yifan Pu, Chao Deng, Junlan Feng,  
601 Shiji Song, and Gao Huang. Dynamic perceiver for efficient visual recognition. In *ICCV*, 2023a.
- 602 Yizeng Han, Zeyu Liu, Zhihang Yuan, Yifan Pu, Chaofei Wang, Shiji Song, and Gao Huang. Latency-  
603 aware unified dynamic networks for efficient image recognition. *arXiv preprint arXiv:2308.15949*,  
604 2023b.
- 605 Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks.  
606 In *Proceedings of the IEEE international conference on computer vision*, pp. 1389–1397, 2017.
- 607 Charles Herrmann, Richard Strong Bowen, and Ramin Zabih. Channel selection using gumbel  
608 softmax. In *ECCV*, pp. 241–257. Springer, 2020.
- 609 Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans  
610 trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural  
611 information processing systems*, 30, 2017.
- 612 Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*,  
613 2022.
- 614 Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in  
615 neural information processing systems*, 33:6840–6851, 2020.
- 616 Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic  
617 bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:  
618 9782–9793, 2020.
- 619 Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- 620 Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved  
621 precision and recall metric for assessing generative models. *Advances in neural information  
622 processing systems*, 32, 2019.
- 623 Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Dynamic  
624 slimmable network. In *CVPR*, pp. 8607–8617, 2021.
- 625 Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang,  
626 and Kurt Keutzer. Q-diffusion: Quantizing diffusion models. In *Proceedings of the IEEE/CVF  
627 International Conference on Computer Vision*, pp. 17535–17545, 2023.
- 628 Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches  
629 are what you need: Expediting vision transformers via token reorganizations. *arXiv preprint  
630 arXiv:2202.07800*, 2022.
- 631 Peiyuan Liao, Xiuyu Li, Xihui Liu, and Kurt Keutzer. The artbench dataset: Benchmarking generative  
632 models with artworks. *arXiv preprint arXiv:2206.11404*, 2022.
- 633 Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr  
634 Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–  
635 ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings,  
636 Part V 13*, pp. 740–755. Springer, 2014.
- 637 I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- 638 Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast  
639 solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*,  
640 2022.

- 648 Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models:  
649 Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*,  
650 2023.
- 651 Xin Ma, Yaohui Wang, Gengyun Jia, Xinyuan Chen, Ziwei Liu, Yuan-Fang Li, Cunjian Chen, and  
652 Yu Qiao. Latte: Latent diffusion transformer for video generation. *arXiv preprint arXiv:2401.03048*,  
653 2024.
- 654 Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free.  
655 *arXiv preprint arXiv:2312.00858*, 2023.
- 656 Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim  
657 Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference*  
658 *on Computer Vision and Pattern Recognition*, pp. 14297–14306, 2023.
- 659 Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-  
660 Nam Lim. Adavit: Adaptive vision transformers for efficient image recognition. In *CVPR*, pp.  
661 12309–12318, 2022.
- 662 Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh,  
663 Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint*  
664 *arXiv:2104.08378*, 2021.
- 665 Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional  
666 neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.
- 667 Taehong Moon, Moonseok Choi, EungGu Yun, Jongmin Yoon, Gayoung Lee, and Juho Lee. Early  
668 exiting for accelerated inference in diffusion models. In *ICML 2023 Workshop on Structured*  
669 *Probabilistic Inference & Generative Modeling*, 2023.
- 670 Taehong Moon, Moonseok Choi, EungGu Yun, Jongmin Yoon, Gayoung Lee, Jaewoong Cho, and  
671 Juho Lee. A simple early exiting framework for accelerated sampling in diffusion models. *arXiv*  
672 *preprint arXiv:2408.05927*, 2024.
- 673 Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W Battaglia. Generating images with  
674 sparse representations. *arXiv preprint arXiv:2103.03841*, 2021.
- 675 Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models.  
676 In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.
- 677 Zizheng Pan, Bohan Zhuang, De-An Huang, Weili Nie, Zhiding Yu, Chaowei Xiao, Jianfei Cai,  
678 and Anima Anandkumar. T-stitch: Accelerating sampling in pre-trained diffusion models with  
679 trajectory stitching. *arXiv preprint arXiv:2402.14167*, 2024.
- 680 William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of*  
681 *the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.
- 682 Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual  
683 reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial*  
684 *intelligence*, volume 32, 2018.
- 685 Jeff Pool and Chong Yu. Channel permutations for n: m sparsity. *Advances in neural information*  
686 *processing systems*, 34:13316–13327, 2021.
- 687 Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit:  
688 Efficient vision transformers with dynamic token sparsification. *NeurIPS*, 34:13937–13949, 2021.
- 689 Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-  
690 resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF confer-*  
691 *ence on computer vision and pattern recognition*, pp. 10684–10695, 2022.
- 692 Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical  
693 image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI*  
694 *2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III*  
695 *18*, pp. 234–241. Springer, 2015.

- 702 Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv*  
703 *preprint arXiv:2202.00512*, 2022.
- 704  
705 Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen.  
706 Improved techniques for training gans. *Advances in neural information processing systems*, 29,  
707 2016.
- 708 Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on  
709 diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern*  
710 *Recognition*, pp. 1972–1981, 2023.
- 711 Junhyuk So, Jungwon Lee, Daehyun Ahn, Hyungjun Kim, and Eunhyeok Park. Temporal dynamic  
712 quantization for diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.
- 713  
714 Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv*  
715 *preprint arXiv:2010.02502*, 2020a.
- 716 Lin Song, Songyang Zhang, Songtao Liu, Zeming Li, Xuming He, Hongbin Sun, Jian Sun, and  
717 Nanning Zheng. Dynamic grained encoder for vision transformers. *NeurIPS*, 34:5770–5783, 2021.
- 718  
719 Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben  
720 Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint*  
721 *arXiv:2011.13456*, 2020b.
- 722 Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint*  
723 *arXiv:2303.01469*, 2023.
- 724 Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via  
725 early exiting from deep neural networks. In *ICPR*, pp. 2464–2469. IEEE, 2016.
- 726  
727 Yao Teng, Yue Wu, Han Shi, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Dim:  
728 Diffusion mamba for efficient high-resolution image synthesis. *arXiv preprint arXiv:2405.14224*,  
729 2024.
- 730 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz  
731 Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing*  
732 *systems*, 30, 2017.
- 733 Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd  
734 birds-200-2011 dataset. 2011.
- 735  
736 Kafeng Wang, Jianfei Chen, He Li, Zhenpeng Mi, and Jun Zhu. Sparsedm: Toward sparse efficient  
737 diffusion models. *arXiv preprint arXiv:2404.10445*, 2024.
- 738 Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic  
739 routing in convolutional networks. In *Proceedings of the European conference on computer vision*  
740 *(ECCV)*, pp. 409–424, 2018.
- 741  
742 Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16  
743 words: Dynamic transformers for efficient image recognition. *NeurIPS*, 34:11960–11973, 2021.
- 744 Enze Xie, Lewei Yao, Han Shi, Zhili Liu, Daquan Zhou, Zhaoqiang Liu, Jiawei Li, and Zhenguo  
745 Li. Diffit: Unlocking transferability of large diffusion models via simple parameter-efficient  
746 fine-tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp.  
747 4230–4239, 2023.
- 748 Jing Nathan Yan, Jiatao Gu, and Alexander M Rush. Diffusion models without attention. In  
749 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.  
750 8239–8249, 2024.
- 751  
752 Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks  
753 for efficient inference. In *CVPR*, pp. 2369–2378, 2020.
- 754 Xingyi Yang, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Diffusion probabilistic model made  
755 slim. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*,  
pp. 22552–22562, 2023.

756 We organize our appendix as follows.  
757

758 **Experimental settings:**

- 759 • Section [A.1](#): Training details of DyDiT on ImageNet.
- 760 • Section [A.2](#): Model configurations of both DiT and DyDiT.
- 761 • Section [A.3](#): Implement details of pruning methods on ImageNet.
- 762 • Section [A.4](#): Details of in-domain fine-tuning on fine-grained datasets.
- 763 • Section [A.5](#): Details of cross-domain fine-tuning.

764  
765  
766 **Additional results:**

- 767 • Section [B.1](#): The inference speed of DyDiT and its acceleration over DiT across models of  
768 varying sizes and specified FLOP budgets.
- 769 • Section [B.2](#): The generalization capability of our method on the U-ViT ([Bao et al., 2023](#))  
770 architecture.
- 771 • Section [B.3](#): Further fine-tuning the original DiT to show that the competitive performance  
772 of our method is not due to the additional fine-tuning.
- 773 • Section [B.4](#): The effectiveness of DyDiT on  $512 \times 512$  resolution image generation.
- 774 • Section [B.5](#): The effectiveness of DyDiT in text-to-image generation, based on PixArt ([Chen  
775 et al., 2023](#)).
- 776 • Section [B.6](#): Integration of DyDiT with a representative distillation-based efficient sampler,  
777 the latent consistency model (LCM) ([Luo et al., 2023](#)).
- 778 • Section [B.7](#): Comparison between DyDiT with the early exiting diffusion model ([Moon  
779 et al., 2023](#)).
- 780 • Section [B.8](#): Fine-tuning efficiency of DyDiT. We fine-tune our model by fewer iterations.
- 781 • Section [B.9](#): Data efficiency of DyDiT. Our model is fine-tuned on only 10% of the training  
782 data.

783 **Visualization:**

- 784 • Section [C.1](#): Additional visualizations of loss maps of DiT-XL.
  - 785 • Section [C.2](#): Additional visualizations of computational cost across different image patches.
  - 786 • Section [C.3](#): Visualization of images generated by DyDiT-XL $_{\lambda=0.5}$  on the ImageNet dataset  
787 at at resolution of  $256 \times 256$ .
- 788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809

## 810 A EXPERIMENTAL SETTINGS.

### 811 A.1 TRAINING DETAILS OF DYDIT ON IMAGENET

812 In Table 6, we present the training details of our model on ImageNet. For DiT-XL, which is pre-  
 813 trained over 7,000,000 iterations, only 200,000 additional fine-tuning iterations (around 3%) are  
 814 needed to enable the dynamic architecture ( $\lambda = 0.5$ ) with our method. For a higher target FLOPs  
 815 ratio  $\lambda = 0.7$ , the iterations can be further reduced.

818 model	DiT-S	DiT-B	DiT-XL
819 optimizer	AdamW (Loshchilov, 2017), learning rate=1e-4		
820 global batch size	256		
821 target FLOPs ratio $\lambda$	[0.9, 0.8, 0.7, 0.5, 0.4, 0.3]	[0.9, 0.8, 0.7, 0.5, 0.4, 0.3]	[0.7, 0.6, 0.5, 0.3]
822 fine-tuning iterations	50,000	100,000	150,000 for $\lambda = 0.7$ 200,000 for others
823 warmup iterations	0	0	30,000
824 augmentation	random flip		
825 cropping size	224×224		

826 Table 6: Experimental settings of our adaption framework.

### 827 A.2 DETAILS OF DiT AND DYDiT MODELS

828 We present the configuration details of the DiT and DyDiT models in Table 7. For DiT-XL, we use  
 829 the checkpoint from the official DiT repository<sup>1</sup> Pan et al. (2024). For DiT-S and DiT-B, we leverage  
 830 pre-trained models from a third-party repository<sup>2</sup> provided by Pan et al. (2024).

831 Table 7: **Details of DiT and DyDiT models.** The router in DyDiT introduce a small number of  
 832 parameters. † denotes that the architecture is dynamically adjusted during generation.

833 model	params. (M) ↓	layers	heads	channel	pre-training	source
834 DiT-S	33	12	6	384	5M iter	Pan et al. (2024)
835 DiT-B	130	12	12	768	1.6M iter	Pan et al. (2024)
836 DiT-XL	675	28	16	1152	7M iter	Peebles & Xie (2023)
837 DyDiT-S	33	12	6 †	384 †	-	-
838 DyDiT-B	131	12	12 †	768 †	-	-
839 DyDiT-XL	678	28	16 †	1152 †	-	-

### 840 A.3 COMPARISON WITH PRUNING METHODS ON IMAGENET.

841 We compare our method with structure pruning and token pruning methods on ImageNet dataset.

- 842 • Random pruning, Magnitude Pruning (He et al., 2017), Taylor Pruning (Molchanov et al.,  
 843 2016), and Diff Pruning (Fang et al., 2024): We adopt the corresponding pruning strategy  
 844 to rank the importance of heads in multi-head self-attention blocks and channels in MLP  
 845 blocks. Then, we prune the least important 50% of heads and channels. The pruned model  
 846 is then fine-tuned for the same number of iterations as its DyDiT counterparts.
- 847 • ToMe (Bolya & Hoffman, 2023): Originally designed to accelerate transformer blocks in  
 848 the U-Net architecture, ToMe operates by merging tokens before the attention block and  
 849 then unmerging them after the MLP blocks. We set the token merging ratio to 20% in each  
 850 block.

### 851 A.4 IN-DOMAIN FINE-TUNING ON FINE-TRAINED DATASETS.

852 We first fine-tune a DiT-S model, which is initialized with parameters pre-trained on ImageNet, on  
 853 a fine-grained dataset. Following the approach in (Xie et al., 2023), we set the training iteration to

854 <sup>1</sup><https://github.com/facebookresearch/DiT>

855 <sup>2</sup><https://github.com/NVlabs/T-Stitch>



24,000. Then, we further fine-tune the model on the same dataset by another 24,000 iterations to adapt the pruning or dynamic architecture to improve the efficiency of the model on the same dataset. We also conduct the generation at a resolution of  $224 \times 224$ . We search optimal classifier-free guidance weights for these methods.

- Random pruning, Magnitude Pruning (He et al., 2017), Taylor Pruning (Molchanov et al., 2016), and Diff Pruning (Fang et al., 2024): For each method, we rank the importance of heads in multi-head self-attention blocks and channels in MLP blocks, pruning the least important 50%.
- ToMe (Bolya & Hoffman, 2023): Originally designed to accelerate transformer blocks in the U-Net architecture, ToMe operates by merging tokens before the attention block and then unmerging them after the MLP blocks. We set the token merging ratio to 20% in each block.

#### A.5 CROSS-DOMAIN TRANSFER LEARNING

In contrast to the aforementioned in-domain fine-tuning, which learns the dynamic strategy within the same dataset, this experiment employs cross-domain fine-tuning. We fine-tune a DiT-S model (pre-trained exclusively on ImageNet) to adapt to the target dataset while simultaneously learning the dynamic architecture. The model is fine-tuned over 48,000 iterations with a batch size of 256.

## B ADDITIONAL RESULTS

### B.1 INFERENCE ACCELERATION.

In Table 8, we present the acceleration ratio of DyDiT compared to the original DiT across different FLOPs targets  $\lambda$ . The results demonstrate that our method effectively enhances batched inference speed, distinguishing our approach from traditional dynamic networks (Herrmann et al., 2020; Meng et al., 2022; Han et al., 2023b), which adapt inference graphs on a per-sample basis and struggle to improve practical efficiency in batched inference.

Table 8: We conduct batched inference on an NVIDIA V100 32G GPU using the optimal batch size for each model. The actual FLOPs of DyDiT may fluctuate around the target FLOPs ratio.

model	s/image ↓	acceleration ↑	FLOPs (G) ↓	FID ↓	FID Δ ↓
DiT-S	0.65	1.00 ×	6.07	21.46	+0.00
DyDiT-S $_{\lambda=0.9}$	0.63	1.03 ×	5.72	21.06	-0.40
DyDiT-S $_{\lambda=0.8}$	0.56	1.16 ×	4.94	21.95	+0.49
DyDiT-S $_{\lambda=0.7}$	0.51	1.27 ×	4.34	23.01	+1.55
DyDiT-S $_{\lambda=0.5}$	0.42	1.54 ×	3.16	28.75	+7.29
DyDiT-S $_{\lambda=0.4}$	0.38	1.71 ×	2.63	36.21	+14.75
DyDiT-S $_{\lambda=0.3}$	0.32	2.03 ×	1.96	59.28	+37.83
DiT-B	2.09	1.00 ×	23.02	9.07	+0.00
DyDiT-B $_{\lambda=0.9}$	1.97	1.05 ×	21.28	8.78	-0.29
DyDiT-B $_{\lambda=0.8}$	1.76	1.18 ×	18.53	8.79	-0.28
DyDiT-B $_{\lambda=0.7}$	1.57	1.32 ×	16.28	9.40	+0.33
DyDiT-B $_{\lambda=0.5}$	1.22	1.70 ×	11.90	12.92	+3.85
DyDiT-B $_{\lambda=0.4}$	1.06	1.95 ×	9.71	15.54	+6.47
DyDiT-B $_{\lambda=0.3}$	0.89	2.33 ×	7.51	23.34	+14.27
DiT-XL	10.22	1.00 ×	118.69	2.27	+0.00
DyDiT-XL $_{\lambda=0.7}$	7.76	1.32 ×	84.33	2.12	-0.15
DyDiT-XL $_{\lambda=0.6}$	6.86	1.49 ×	67.83	2.18	-0.09
DyDiT-XL $_{\lambda=0.5}$	5.91	1.73 ×	57.88	2.07	-0.20
DyDiT-XL $_{\lambda=0.3}$	4.26	2.40 ×	38.85	3.36	+1.09

## B.2 EFFECTIVENESS ON U-ViT.

We evaluate the architecture generalization capability of our method through experiments on U-ViT (Bao et al., 2023), a transformer-based diffusion model with skip connections similar to U-Net (Ronneberger et al., 2015). The results, shown in Table 9, indicate that configuring the target FLOPs ratio  $\lambda$  to 0.4 and adapting U-ViT-S/2 to our dynamic architecture (denoted as DyUViT-S/2  $\lambda=0.4$ ) reduces computational cost from 11.34 GFLOPs to 4.73 GFLOPs, while maintaining a comparable FID score. We also compare our method with the structure pruning method Diff Pruning (Fang et al., 2024) and sparse pruning methods ASP (Pool & Yu, 2021; Mishra et al., 2021) and SparseDM (Wang et al., 2024). The results verify the superiority of our dynamic architecture over static pruning.

In Table 10, we apply our method to the largest model, U-ViT-H/2, and conduct experiments on ImageNet. The results demonstrate that our method effectively accelerates U-ViT-H/2 with only a marginal performance drop. These results verify the generalizability of our method in U-ViT.

Table 9: **U-ViT (Bao et al., 2023) performs image generation on the CIFAR-10 dataset (Krizhevsky et al., 2009)**. Aligning with its default configuration, we generate images using 1,000 diffusion steps with the Euler-Maruyama SDE sampler (Song et al., 2020b).

model	s/image ↓	acceleration ↑	FLOPs (G) ↓	FID ↓	FID Δ ↓
U-ViT-S/2	2.19	1.00 ×	11.34	3.12	0.00
DyU-ViT-S/2 $\lambda=0.4$	1.04	2.10 ×	4.73	3.18	+0.06
pruned w/ Diff	-	-	5.32	12.63	+9.51
pruned w/ ASP	-	-	5.76	319.87	+316.75
pruned w/ SparseDM	-	-	5.67	4.23	+1.11

Table 10: **U-ViT (Bao et al., 2023) performs image generation on the ImageNet (Deng et al., 2009)**. Aligning with its default configuration, we generate images using 50-step DPM-solver++ (Lu et al., 2022).

model	s/image ↓	acceleration ↑	FLOPs (G) ↓	FID ↓	FID Δ ↓
U-ViT-H/2	2.22	1.00 ×	113.00	2.29	0.00
DyU-ViT-H/2 $\lambda=0.5$	1.35	1.57 ×	67.09	2.42	+0.13

## B.3 FURTHER FINE-TUNE ORIGINAL DiT ON IMAGENET.

our method is not attributed to additional fine-tuning. In Table 11, we fine-tune the original DiT for 150,000 and 350,000 iterations, observing a slight improvement in the FID score, which fluctuates around 2.16. Under the same iterations, DyDiT achieves a better FID while significantly reducing FLOPs, verifying that the improvement is due to our design rather than extended training iterations.

Table 11: **Further fine-tuning original DiT on ImageNet.**

model	pre-trained iterations	fine-tuning iterations	FLOPs (G) ↓	FID ↓	FID Δ ↓
DiT-XL	7,000,000	-	118.69	2.27	+0.00
DiT-XL	7,000,000	150,000 (2.14%)	118.69	2.16	-0.11
DiT-XL	7,000,000	350,000 (5.00%)	118.69	2.15	-0.12
DyDiT-XL $\lambda=0.7$	7,000,000	150,000 (2.14%)	84.33	2.12	-0.15

## B.4 EFFECTIVENESS IN HIGH-RESOLUTION GENERATION.

We conduct experiments to generate images at a resolution of  $512 \times 512$  to validate the effectiveness of our method for high-resolution generation. We use the official checkpoint of DiT-XL  $512 \times 512$  as the baseline, which is trained on ImageNet (Deng et al., 2009) for 3,000,000 iterations. We fine-tune it for 150,000 iterations to enable its dynamic architecture, denoted as DyDiT-XL  $512 \times 512$ . The

target FLOP ratio is set to 0.7. The experimental results, presented in Table 12, demonstrate that our method achieves a superior FID score compared to the original DiT-XL, while requiring fewer FLOPs.

Table 12: **Image generation at 512×512 resolution on ImageNet (Deng et al., 2009)**. We sample 50,000 images and leverage FID to measure the generation quality. We adopt 100 DDPM steps to generate images.

model	s/image ↓	acceleration ↑	FLOPs (G) ↓	FID ↓	FID Δ ↓
DiT-XL 512×512	18.36	1.00×	514.80	3.75	0.00
DyDiT-XL 512×512 $\lambda=0.7$	14.00	1.31×	375.35	3.61	-0.14

### B.5 EFFECTIVENESS IN TEXT-TO-IMAGE GENERATION.

We further validate the applicability of our method in text-to-image generation, which is more challenging than the class-to-image generation. We adopt PixArt- $\alpha$  (Chen et al., 2023), a text-to-image generation model built based on DiT (Peebles & Xie, 2023) as the baseline. PixArt- $\alpha$  is pre-trained on extensive private datasets and exhibits superior text-to-image generation capabilities. Our model is initialized using the official PixArt- $\alpha$  checkpoint fine-tuned on the COCO dataset (Lin et al., 2014). We further fine-tune it with our method to enable dynamic architecture adaptation, resulting in the DyPixArt- $\alpha$  model, as shown in Table 13. Notably, DyPixArt- $\alpha$  with  $\lambda = 0.7$  achieves an FID score comparable to the original PixArt- $\alpha$ , while significantly accelerating the generation.

Table 13: **Text-to-image generation on COCO (Lin et al., 2014)**. We randomly select text prompts from COCO and adopt 20-step DPM-solver++ (Lu et al., 2022) to sample 30,000 images for evaluating the FID score.

Model	s/image ↓	acceleration ↑	FLOPs (G) ↓	FID ↓	FID Δ ↓
PixArt- $\alpha$	0.91	1.00 ×	141.09	19.88	+0.00
DyPixArt- $\alpha_{\lambda=0.7}$	0.69	1.32 ×	112.44	19.75	-0.13

### B.6 EXPLORATION OF COMBINING LCM WITH DYDiT .

Some sampler-based efficient methods (Meng et al., 2023; Song et al., 2023; Luo et al., 2023) adopt distillation techniques to reduce the generation process to several steps. In this section, we combine our DyDiT, a model-based method, with a representative method, the latent consistency model (LCM) (Luo et al., 2023) to explore their compatibility for superior generation speed. In LCM, the generation process can be reduced to 1-4 steps via consistency distillation and the 4-step generation achieves an satisfactory balance between performance and efficiency. Hence, we conduct experiments in the 4-step setting. Under the target FLOPs ratio  $\lambda = 0.9$ , our method further accelerates generation and achieves comparable performance, demonstrating its potential with LCM. However, further reducing the FLOPs ratio leads to model collapse. This issue may arise because DyDiT’s training depends on noise prediction difficulty, which is absent in LCM distillation, causing instability at lower FLOPs ratios. This encourage us to develop dynamic models and training strategies for distillation-based efficient samplers to achieve superior generation efficiency in the future.

Table 14: **Combining DyDiT with Latent Consistency Model (LCM) (Luo et al., 2023)** . We conduct experiments under the 4-step LCM setting, as it achieves a satisfactory balance between performance and efficiency.

model	s/image ↓	FLOPs (G) ↓	FID ↓	FID Δ ↓
DiT-XL+250-step DDPM	10.22	118.69	2.27	+0.00
DiT-XL + 4-step LCM	0.082	118.69	6.53	+4.26
DyDiT-XL $\lambda=0.9$ + 4-step LCM	0.076	104.43	6.52	+4.25

## B.7 COMPARISON WITH THE EARLY EXITING METHOD.

We compare our approach with the early exiting diffusion model ASE (Moon et al., 2023; 2024), which implements a strategy to selectively skip layers for certain timesteps. Following their methodology, we evaluate the FID score using 5,000 samples. Results are summarized in Table 15. Despite similar generation performance, our method achieves a better acceleration ratio, demonstrating the effectiveness of our design.

Table 15: **Comparison with the early exiting method (Moon et al., 2023; 2024).** As methods may be evaluated on different devices, we report only the acceleration ratio for speed comparison.

model	acceleration $\uparrow$	FID $\downarrow$	FID $\Delta \downarrow$
DiT-XL	1.00 $\times$	9.08	0.00
DyDiT-XL $_{\lambda=0.5}$	1.73 $\times$	8.95	-0.13
ASE-D4 DiT-XL	1.34 $\times$	9.09	+0.01
ASE-D7 DiT-XL	1.39 $\times$	9.39	+0.31

## B.8 TRAINING EFFICIENCY

Our approach enhances the inference efficiency of the diffusion transformer while maintaining training efficiency. It requires only a small number of additional fine-tuning iterations to learn the dynamic architecture. In Table 16, we present our model with various fine-tuning iterations and their corresponding FID scores. The original DiT-XL model is pre-trained on the ImageNet dataset over 7,000,000 iterations with a batch size of 256. Remarkably, our method achieves a 2.12 FID score with just 50,000 fine-tuning iterations to adopt the dynamic architecture—approximately 0.7% of the pre-training schedule. Furthermore, when extended to 100,000 and 150,000 iterations, our method performs comparably to DiT. We observe that the actual FLOPs during generation converge as the number of fine-tuning iterations increases.

Table 16: **Training efficiency.** The original DiT-XL model is pre-trained on the ImageNet dataset over 7,000,000 iterations with a batch size of 256.

model	fine-tuning iterations	FLOPs (G) $\downarrow$	FID $\downarrow$	FID $\Delta \downarrow$
DiT-XL	-	118.69	2.27	+0.00
DyDiT-XL $_{\lambda=0.7}$	10,000 (0.14%)	103.08	45.95	43.65
DyDiT-XL $_{\lambda=0.7}$	25,000 (0.35%)	91.97	2.97	+0.70
DyDiT-XL $_{\lambda=0.7}$	50,000 (0.71%)	85.07	2.12	-0.15
DyDiT-XL $_{\lambda=0.7}$	100,000 (1.43%)	84.30	2.17	-0.10
DyDiT-XL $_{\lambda=0.7}$	150,000 (2.14%)	84.33	2.12	-0.15

## B.9 DATA EFFICIENCY

To evaluate the data efficiency of our method, we randomly sampled 10% of the ImageNet dataset (Deng et al., 2009) for training. DyDiT was fine-tuned on this subset to adapt the dynamic architecture. As shown in Table 17, when fine-tuned on just 10% of the data, our model DyDiT-XL $_{\lambda=0.7}$  still achieves performance comparable to the original DiT. When we further reduce the fine-tuning data ratio to 1%, the FID score increase slightly by 0.06. These results indicate that our method maintains robust performance even with limited fine-tuning data.

## C VISUALIZATION

### C.1 ADDITIONAL VISUALIZATION OF LOSS MAPS

In Figure 7, we visualize the loss maps (normalized to the range [0, 1]) for several timesteps, demonstrating that noise in different image patches exhibits varying levels of prediction difficulty.

Table 17: **Data efficiency.** The slight difference in FLOPs of our models is introduced by the learned TDW and SDT upon fine-tuning convergence.

model	fine-tuning data ratio	FLOPs (G) ↓	FID ↓	FID $\Delta$ ↓
DiT-XL	-	118.69	2.27	+0.00
DyDiT-XL $_{\lambda=0.7}$	100%	84.33	2.12	-0.15
DyDiT-XL $_{\lambda=0.7}$	10%	84.43	2.13	-0.14
DyDiT-XL $_{\lambda=0.7}$	1%	84.37	2.31	+0.06

## C.2 ADDITIONAL VISUALIZATION OF COMPUTATIONAL COST ON IMAGE PATCHES

In Figure 8, we quantify and normalize the computational cost across different image patches during generation, ranging from [0, 1]. The proposed spatial-wise dynamic token strategy learns to adjust the computational cost for each image patch.

## C.3 VISUALIZATION OF SAMPLES FROM DYDiT-XL

We visualize the images generated by DyDiT-XL $_{\lambda=0.5}$  on the ImageNet (Deng et al., 2009) dataset at a resolution of  $256 \times 256$  from Figure 9 to Figure 22. The classifier-free guidance scale is set to 4.0. All samples here are uncurated.

1134  
 1135  
 1136  
 1137  
 1138  
 1139  
 1140  
 1141  
 1142  
 1143  
 1144  
 1145  
 1146  
 1147  
 1148  
 1149  
 1150  
 1151  
 1152  
 1153  
 1154  
 1155  
 1156  
 1157  
 1158  
 1159  
 1160  
 1161  
 1162  
 1163  
 1164  
 1165  
 1166  
 1167  
 1168  
 1169  
 1170  
 1171  
 1172  
 1173  
 1174  
 1175  
 1176  
 1177  
 1178  
 1179  
 1180  
 1181  
 1182  
 1183  
 1184  
 1185  
 1186  
 1187

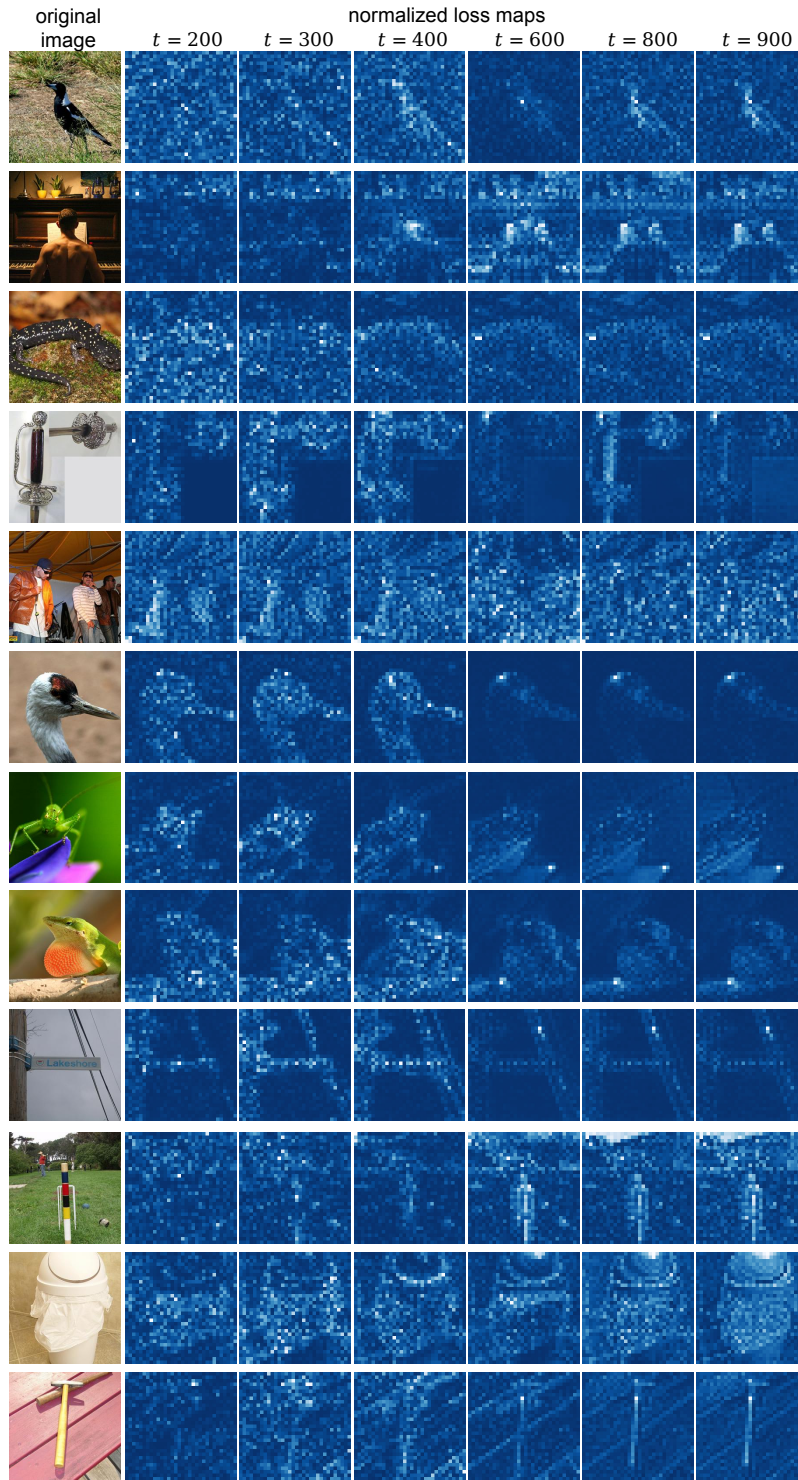


Figure 7: **Additional visualization of loss maps from DiT-XL.** The loss values are normalized to the range [0, 1]. Different image patches exhibit varying levels of prediction difficulty.

1188  
 1189  
 1190  
 1191  
 1192  
 1193  
 1194  
 1195  
 1196  
 1197  
 1198  
 1199  
 1200  
 1201  
 1202  
 1203  
 1204  
 1205  
 1206  
 1207  
 1208  
 1209  
 1210  
 1211  
 1212  
 1213  
 1214  
 1215  
 1216  
 1217  
 1218  
 1219  
 1220  
 1221  
 1222  
 1223  
 1224  
 1225  
 1226  
 1227  
 1228  
 1229  
 1230  
 1231  
 1232  
 1233  
 1234  
 1235  
 1236  
 1237  
 1238  
 1239  
 1240  
 1241

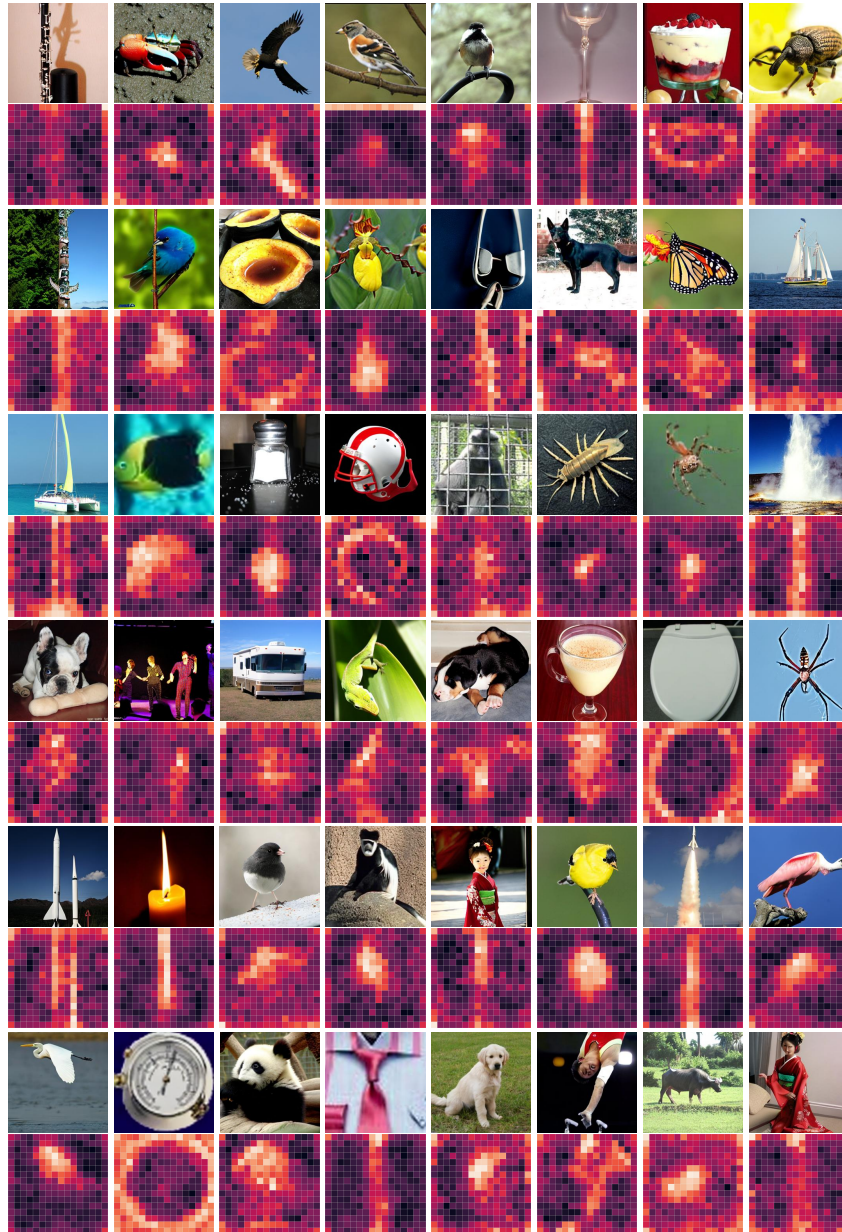


Figure 8: **Additional visualizations of computational cost across different image patches.** Complementary to Figure 6, we visualize more generated images and their corresponding FLOPs cost across different image patches. The map is normalized to  $[0, 1]$  for clarity.

1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268

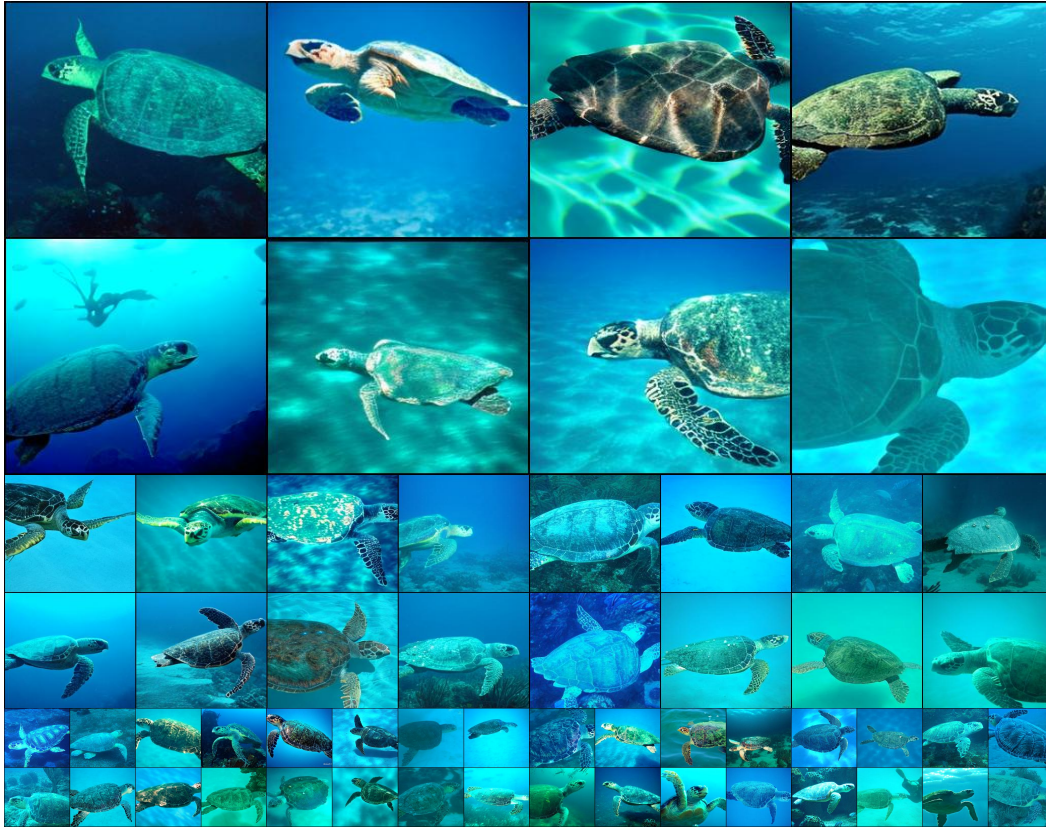


Figure 9: Uncurated  $256 \times 256$  DyDiT-XL $_{\lambda=0.5}$  samples. Loggerhead turtle (33).

1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295



Figure 10: Uncurated  $256 \times 256$  DyDiT-XL $_{\lambda=0.5}$  samples. Macaw (88).



1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322



Figure 11: Uncurated  $256 \times 256$  DyDiT-XL $_{\lambda=0.5}$  samples. Kakatoe galerita (89).

1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349



Figure 12: Uncurated  $256 \times 256$  DyDiT-XL $_{\lambda=0.5}$  samples. Golden retriever (207).

1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376



Figure 13: Uncurated  $256 \times 256$  DyDiT-XL $_{\lambda=0.5}$  samples. Siberian husky (250).

1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403



Figure 14: Uncurated  $256 \times 256$  DyDiT-XL $_{\lambda=0.5}$  samples. Lion (291).

1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430



Figure 15: Uncurated  $256 \times 256$  DyDiT-XL $_{\lambda=0.5}$  samples. Lesser panda(387).

1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457



Figure 16: Uncurated  $256 \times 256$  DyDiT-XL $_{\lambda=0.5}$  samples. Panda (388).

1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511

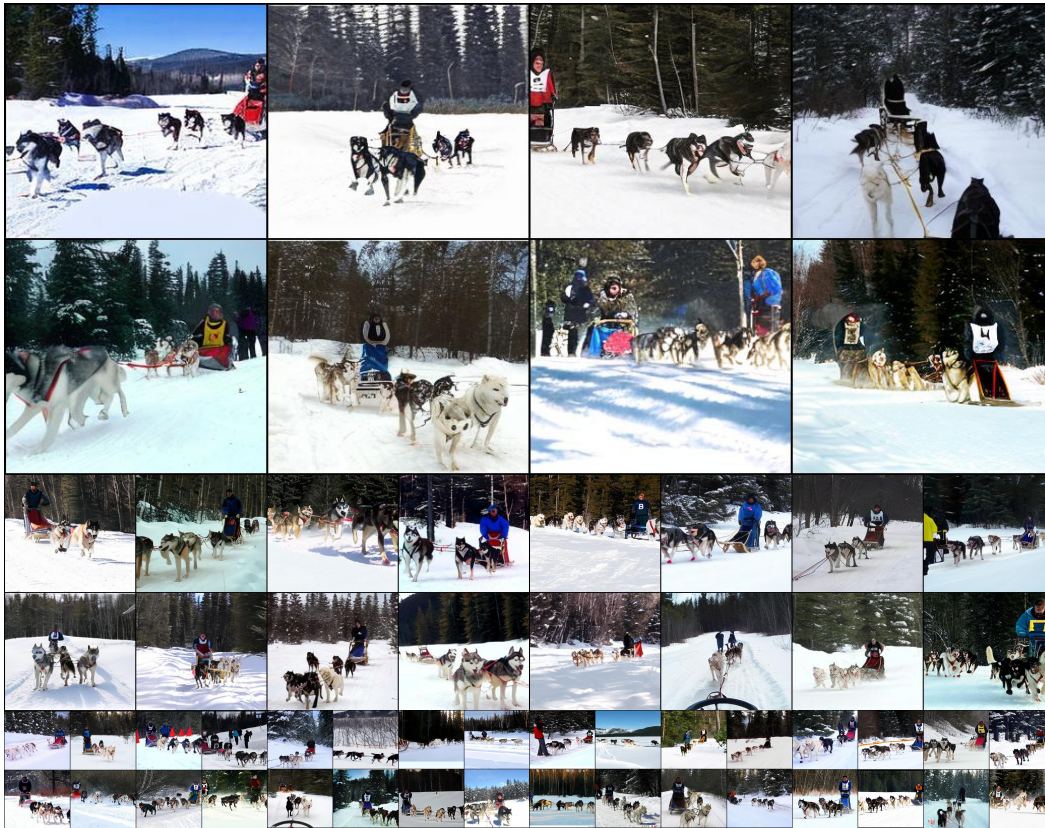


Figure 17: Uncurated  $256 \times 256$  DyDiT-XL $_{\lambda=0.5}$  samples. Dogsled (537).



Figure 18: Uncurated  $256 \times 256$  DyDiT-XL $_{\lambda=0.5}$  samples. Space shuttle (812).

1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538



Figure 19: Uncurated  $256 \times 256$  DyDiT-XL $_{\lambda=0.5}$  samples. Ice cream (928).

1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565

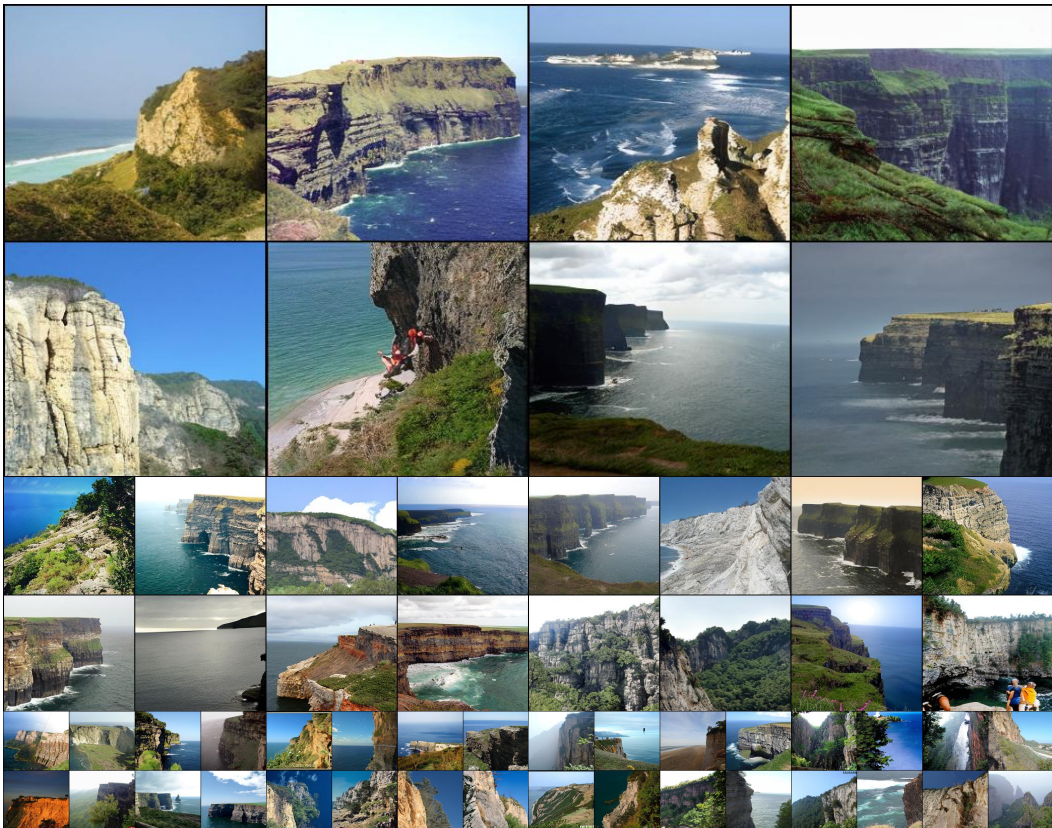


Figure 20: Uncurated  $256 \times 256$  DyDiT-XL $_{\lambda=0.5}$  samples. cliff(972).



Figure 21: Uncurated  $256 \times 256$  DyDiT-XL $_{\lambda=0.5}$  samples. Lakeside (975).



Figure 22: Uncurated  $256 \times 256$  DyDiT-XL $_{\lambda=0.5}$  samples. Volcano (980).