TinyFusion: Diffusion Transformers Learned Shallow

Gongfan Fang*, Kunjun Li*, Xinyin Ma, Xinchao Wang†
National University of Singapore
{gongfan, kunjun, maxinyin}@u.nus.edu, xinchao@nus.edu.sg

Abstract

Diffusion Transformers have demonstrated remarkable capabilities in image generation but often come with excessive parameterization, resulting in considerable inference overhead in real-world applications. In this work, we present TinyFusion, a depth pruning method designed to remove redundant layers from diffusion transformers via end-to-end learning. The core principle of our approach is to create a pruned model with high recoverability*, allowing it to regain strong performance after fine-tuning. To accomplish this, we introduce a differentiable sampling technique to make pruning learnable, paired with a co-optimized parameter to simulate future fine-tuning. While prior works focus on minimizing loss or error after pruning, our method explicitly models and optimizes the post-fine-tuning performance of pruned models. Experimental results indicate that this learnable paradigm offers substantial benefits for layer pruning of diffusion transformers, surpassing existing importance-based and error-based methods. Additionally, TinyFusion exhibits strong generalization across diverse architectures, such as DiTs, MARs, and SiTs. Experiments with DiT-XL show that TinyFusion can craft a shallow diffusion transformer at less than 7% of the pre-training cost, achieving a 2× speedup with an FID score of 2.86, outperforming competitors with comparable efficiency. Code is available at* `https://github.com/VainF/TinyFusion`

1. Introduction

Diffusion Transformers have emerged as a cornerstone architecture for generative tasks, achieving notable success in areas such as image [11, 26, 40] and video synthesis [25, 59]. This success has also led to the widespread availability of high-quality pre-trained models on the Internet, greatly accelerating and supporting the development of various downstream applications [5, 16, 53, 55]. However, pre-trained diffusion transformers usually come with considerable inference costs due to the huge parameter scale, which poses significant challenges for deployment. To resolve this problem, there has been growing interest from both the research community and industry in developing lightweight models [12, 23, 32, 58].

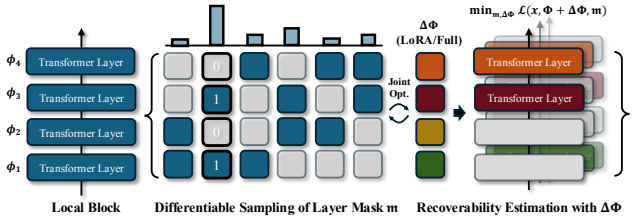


Figure 1. This work presents a learnable approach for pruning the depth of pre-trained diffusion transformers. Our method simultaneously optimizes a differentiable sampling process of layer masks and a weight update to identify a highly recoverable solution, ensuring that the pruned model maintains competitive performance after fine-tuning.

The efficiency of diffusion models is typically influenced by various factors, including the number of sampling steps [33, 43, 45, 46], operator design [7, 48, 52], computational precision [19, 30, 44], network width [3, 12] and depth [6, 23, 36]. In this work, we focus on model compression through depth pruning [36, 54], which removes entire layers from the network to reduce the latency. Depth pruning offers a significant advantage in practice: it can achieve a linear acceleration ratio relative to the compression rate on both parallel and non-parallel devices. For example, as will be demonstrated in this work, while 50% width pruning [12] only yields a 1.6× speedup, pruning 50% of the layers results in a 2× speedup. This makes depth pruning a flexible and practical method for model compression.

This work follows a standard depth pruning framework: unimportant layers are first removed, and the pruned model is then fine-tuned for performance recovery. In the literature, depth pruning techniques designed for diffusion transformers or general transformers primarily focus on heuristic approaches, such as carefully designed

*Equal contribution: Kunjun contributed equally to the method design, analytical experiments, and manuscript preparation.
†Corresponding author

importance scores [6, 36] or manually configured pruning schemes [23, 54]. These methods adhere to a loss minimization principle [18, 37], aiming to identify solutions that maintain low loss or error after pruning. This paper investigates the effectiveness of this widely used principle in the context of depth compression. Through experiments, we examined the relationship between calibration loss observed post-pruning and the performance after fine-tuning. This is achieved by extensively sampling 100,000 models via random pruning, exhibiting different levels of calibration loss in the searching space. Based on this, we analyzed the effectiveness of existing pruning algorithms, such as the feature similarity [6, 36] and sensitivity analysis [18], which indeed achieve low calibration losses in the solution space. However, the performance of all these models after fine-tuning often falls short of expectations. This indicates that the loss minimization principle may not be well-suited for diffusion transformers.

Building on these insights, we reassessed the underlying principles for effective layer pruning in diffusion transformers. Fine-tuning diffusion transformers is an extremely time-consuming process. Instead of searching for a model that minimizes loss immediately after pruning, we propose identifying candidate models with strong recoverability, enabling superior post-fine-tuning performance. Achieving this goal is particularly challenging, as it requires the integration of two distinct processes, pruning and fine-tuning, which involve non-differentiable operations and cannot be directly optimized via gradient descent.

To this end, we propose a learnable depth pruning method that effectively integrates pruning and fine-tuning. As shown in Figure 1, we model the pruning and fine-tuning of a diffusion transformer as a differentiable sampling process of layer masks [13, 17, 22], combined with a co-optimized weight update to simulate future fine-tuning. Our objective is to iteratively refine this distribution so that networks with higher recoverability are more likely to be sampled. This is achieved through a straightforward strategy: if a sampled pruning decision results in strong recoverability, similar pruning patterns will have an increased probability of being sampled. This approach promotes the exploration of potentially valuable solutions while disregarding less effective ones. Additionally, the proposed method is highly efficient, and we demonstrate that a suitable solution can emerge within a few training steps.

To evaluate the effectiveness of the proposed method, we conduct extensive experiments on various transformer-based diffusion models, including DiTs [40], MARs [29], SiTs [34]. The learnable approach is highly efficient. It is able to identify redundant layers in diffusion transformers with 1-epoch training on the dataset, which effectively crafts shallow diffusion transformers from pre-trained models with high recoverability. For instance, while the models pruned by TinyFusion initially exhibit relatively high calibration loss after removing 50% of layers, they recover quickly through fine-tuning, achieving a significantly more competitive FID score (5.73 vs. 22.28) compared to baseline methods that only minimize immediate loss, using just 1% of the pre-training cost. Additionally, we also explore the role of knowledge distillation in enhancing recoverability [20, 23] by introducing a MaskedKD variant. MaskedKD mitigates the negative impact of the massive or outlier activations [47] in hidden states, which can significantly affect the performance and reliability of fine-tuning. With MaskedKD, the FID score improves from 5.73 to 3.73 with only 1% of pre-training cost. Extending the training to 7% of the pre-training cost further reduces the FID to 2.86, just 0.4 higher than the original model with doubled depth.

Therefore, the main contribution of this work lies in a learnable method to craft shallow diffusion transformers from pre-trained ones, which explicitly optimizes the recoverability of pruned models. The method is general for various architectures, including DiTs, MARs and SiTs.

2. Related Works

Network Pruning and Depth Reduction. Network pruning is a widely used approach for compressing pre-trained diffusion models by eliminating redundant parameters [3, 12, 31, 51]. Diff-Pruning [12] introduces a gradient-based technique to streamline the width of UNet, followed by a simple fine-tuning to recover the performance. SparseDM [51] applies sparsity to pre-trained diffusion models via the Straight-Through Estimator (STE) [2], achieving a 50% reduction in MACs with only a 1.22 increase in FID on average. While width pruning and sparsity help reduce memory overhead, they often offer limited speed improvements, especially on parallel devices like GPUs. Consequently, depth reduction has gained significant attention in the past few years, as removing entire layers enables better speedup proportional to the pruning ratio [24, 27, 28, 36, 54, 56, 58]. Adaptive depth reduction techniques, such as MoD [41] and depth-aware transformers [10], have also been proposed. Despite these advances, most existing methods are still based on empirical or heuristic strategies, such as carefully designed importance criteria [36, 54], sensitivity analyses [18] or manually designed schemes [23], which often do not yield strong performance guarantee after fine-tuning.

Efficient Diffusion Transformers. Developing efficient diffusion transformers has become an appealing focus within the community, where significant efforts have been made to enhance efficiency from various perspectives, including linear attention mechanisms [15, 48, 52], compact architectures [50], non-autoregressive transformers [4, 14, 38, 49], pruning [12, 23], quantization [19, 30, 44], feature

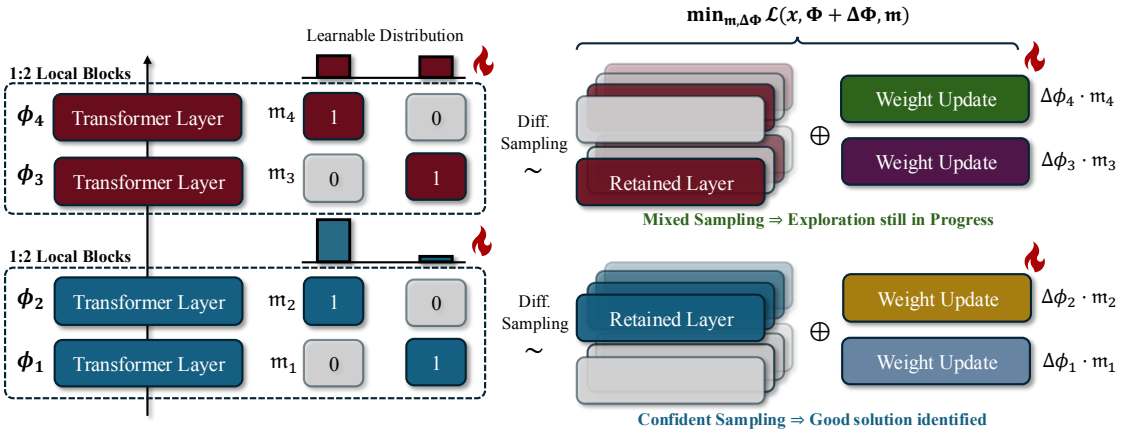


Figure 2. The proposed TinyFusion method learns to perform a differentiable sampling of candidate solutions, jointly optimized with a weight update to estimate recoverability. This approach aims to increase the likelihood of favorable solutions that ensure strong post-fine-tuning performance. After training, local structures with the highest sampling probabilities are retained.

caching [35, 57], etc. In this work, we focus on compressing the depth of pre-trained diffusion transformers and introduce a learnable method that directly optimizes recoverability, which is able to achieve satisfactory results with low re-training costs.

3. Method

3.1. Shallow Generative Transformers by Pruning

This work aims to derive a shallow diffusion transformer by pruning a pre-trained model. For simplicity, all vectors in this paper are column vectors. Consider a L-layer transformer, parameterized by $\Phi_{L\times D} = [\phi_1, \phi_2, \cdots, \phi_L]^\intercal$, where each element ϕ_i encompasses all learnable parameters of a transformer layer as a D-dim column vector, which includes the weights of both attention layers and MLPs. Depth pruning seeks to find a binary layer mask $\mathbf{m}_{L\times 1} = [m_1, m_2, \cdots, m_L]^\intercal$, that removes a layer by:

$$x_{i+1} = m_i\boldsymbol{\phi}_i(x_i) + (1-m_i)x_i = \begin{cases} \boldsymbol{\phi}_i(x_i), & \text{if } m_i = 1, \\ x_i, & \text{otherwise,} \end{cases} \tag{1}$$

where the x_i and $\boldsymbol{\phi}_i(x_i)$ refers to the input and output of layer $\boldsymbol{\phi}_i$. To obtain the mask, a common paradigm in prior work is to minimize the loss $\mathcal{L}$ after pruning, which can be formulated as $\min_{\mathbf{m}} \mathbb{E}_x\left[\mathcal{L}(x, \Phi, \mathbf{m})\right]$. However, as we will show in the experiments, this objective – though widely adopted in discriminative tasks – may not be well-suited to pruning diffusion transformers. Instead, we are more interested in the recoverability of pruned models. To achieve this, we incorporate an additional weight update into the optimization problem and extend the objective by:

$$\min_{\mathbf{m}} \underbrace{\min_{\Delta\Phi} \mathbb{E}_x\left[\mathcal{L}(x, \Phi + \Delta\Phi, \mathbf{m})\right]}_{\textit{Recoverability: Post-Fine-Tuning Performance}}, \tag{2}$$

where $\Delta\Phi = \{\Delta\phi_1, \Delta\phi_2, \cdots, \Delta\phi_M\}$ represents appropriate update from fine-tuning. The objective formulated by Equation 2 poses two challenges: 1) The non-differentiable nature of layer selection prevents direct optimization using gradient descent; 2) The inner optimization over the retained layers makes it computationally intractable to explore the entire search space, as this process necessitates selecting a candidate model and fine-tuning it for evaluation. To address this, we propose TinyFusion that makes both the pruning and recoverability optimizable.

3.2. TinyFusion: Learnable Depth Pruning

A Probabilistic Perspective. This work models Equation 2 from a probabilistic standpoint. We hypothesize that the mask $\mathbf{m}$ produced by "ideal" pruning methods (might be not unique) should follow a certain distribution. To model this, it is intuitive to associate every possible mask $\mathbf{m}$ with a probability value $p(\mathbf{m})$, thus forming a categorical distribution. Without any prior knowledge, the assessment of pruning masks begins with a uniform distribution. However, directly sampling from this initial distribution is highly inefficient due to the vast search space. For instance, pruning a 28-layer model by 50% involves evaluating $\binom{28}{14} = 40,116,600$ possible solutions. To overcome this challenge, this work introduces an advanced and learnable algorithm capable of using evaluation results as feedback to iteratively refine the mask distribution. The basic idea is that if certain masks exhibit positive results, then other masks with *similar pattern* may also be potential solutions and thus should have a higher likelihood of sampling in subsequent evaluations, allowing for a more focused search on promising solutions. However, the definition of "similarity pattern" is still unclear so far.

Sampling Local Structures. In this work, we demonstrate that local structures, as illustrated in Figure 2, can serve as effective anchors for modeling the relationships between different masks. If a pruning mask leads to certain local structures and yields competitive results after fine-tuning, then other masks yielding the same local patterns are also likely to be positive solutions. This can be achieved by dividing the original model into K non-overlapping blocks, represented as $\Phi = [\Phi_1, \Phi_2, \cdots, \Phi_K]^\intercal$. For simplicity, we assume each block $\Phi_k = [\phi_{k1}, \phi_{k2}, \cdots, \phi_{kM}]^\intercal$ contains exactly M layers, although they can have varied lengths. Instead of performing global layer pruning, we propose an N:M scheme for local layer pruning, where, for each block Φ_k with M layers, N layers are retained. This results in a set of local binary masks $\mathbf{m} = [\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K]^\intercal$. Similarly, the distribution of a local mask $\mathbf{m}_k$ is modeled using a categorical distribution $p(\mathbf{m}_k)$. We perform independent sampling of local binary masks and combine them for pruning, which presents the joint distribution:

$$p(\mathbf{m}) = p(\mathbf{m}_1) \cdot p(\mathbf{m}_2) \cdots p(\mathbf{m}_K) \tag{3}$$

If some local distributions $p(\mathbf{m}_k)$ exhibit high confidence in the corresponding blocks, the system will tend to sample those positive patterns frequently and keep active explorations in other local blocks. Based on this concept, we introduce differential sampling to make the above process learnable.

Differentiable Sampling. Considering the sampling process of a local mask $\mathbf{m}_k$, which corresponds a local block Φ_k and is modeled by a categorical distribution $p(\mathbf{m}_k)$. With the N:M scheme, there are $\binom{M}{N}$ possible masks. We construct a special matrix $\hat{\mathbf{m}}^{N:M}$ to enumerate all possible masks. For example, 2:3 layer pruning will lead to the candidate matrix $\hat{\mathbf{m}}^{2:3} = [[1,1,0],[1,0,1],[0,1,1]]$. In this case, each block will have three probabilities $p(\mathbf{m}_k) = [p_{k1}, p_{k2}, p_{k3}]$. For simplicity, we omit $\mathbf{m}_k$ and k and use p_i to represent the probability of sampling i-th element in $\hat{\mathbf{m}}^{N:M}$. A popular method to make a sampling process differentiable is Gumbel-Softmax [13, 17, 22]:

$$y = \text{one-hot}\left(\frac{\exp((g_i + \log p_i)/\tau)}{\sum_j \exp((g_j + \log p_j)/\tau)}\right). \tag{4}$$

where g_i is random noise drawn from the Gumbel distribution $\mathit{Gumbel}(0,1)$ and τ refers to the temperature term. The output y is the index of the sampled mask. Here a Straight-Through Estimator [2] is applied to the one-hot operation, where the onehot operation is enabled during forward and is treated as an identity function during backward. Leveraging the one-hot index y and the candidate set $\hat{\mathbf{m}}^{N:M}$, we can draw a mask $\mathbf{m} \sim p(\mathbf{m})$ through a simple index operation:

$$\mathbf{m} = y^\intercal \hat{\mathbf{m}} \tag{5}$$

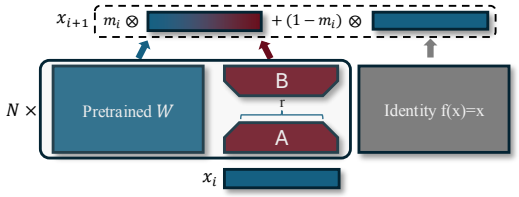


Figure 3. An example of forward propagation with differentiable pruning mask m_i and LoRA for recoverability estimation.

Notably, when $\tau \to 0$, the STE gradients will approximate the true gradients, yet with a higher variance which is negative for training [22]. Thus, a scheduler is typically employed to initiate training with a high temperature, gradually reducing it over time.

Joint Optimization with Recoverability. With differentiable sampling, we are able to update the underlying probability using gradient descent. The training objective in this work is to maximize the recoverability of sampled masks. We reformulate the objective in Equation 2 by incorporating the learnable distribution:

$$\min_{\{p(\mathbf{m}_k)\}} \underbrace{\min_{\Delta\Phi} \mathbb{E}_{x,\{\mathbf{m}_k \sim p(\mathbf{m}_k)\}} \left[\mathcal{L}(x, \Phi + \Delta\Phi, \{\mathbf{m}_k\}\right]}_{\textit{Recoverability: Post-Fine-Tuning Performance}}, \tag{6}$$

where $\{p(\mathbf{m}_k)\} = \{p(\mathbf{m}_1), \cdots, p(\mathbf{m}_K)\}$ refer to the categorical distributions for different local blocks. Based on this formulation, we further investigate how to incorporate the fine-tuning information into the training. We propose a joint optimization of the distribution and a weight update $\Delta\Phi$. Our key idea is to introduce a co-optimized update $\Delta\Phi$ for joint training. A straightforward way to craft the update is to directly optimize the original network. However, the parameter scale in a diffusion transformer is usually huge, and a full optimization may make the training process costly and not that efficient. To this end, we show that Parameter-Efficient Fine-Tuning methods such as LoRA [21] can be a good choice to obtain the required $\Delta\Phi$. For a single linear matrix $\mathbf{W}$ in Φ, we simulate the fine-tuned weights as:

$$\mathbf{W}_{\text{fine-tuned}} = \mathbf{W} + \alpha\Delta\mathbf{W} = \mathbf{W} + \alpha\mathbf{B}\mathbf{A}, \tag{7}$$

where α is a scalar hyperparameter that scales the contribution of $\Delta\mathbf{W}$. Using LoRA significantly reduces the number of parameters, facilitating efficient exploration of different pruning decisions. As shown in Figure 3, we leverage the sampled binary mask value m_i as the gate and forward the network with Equation 1, which suppresses the layer outputs if the sampled mask is 0 for the current layer. In addition, the previously mentioned STE will still provide non-zero gradients to the pruned layer, allowing it to be further updated. This is helpful in practice, since some layers

Method	Depth	#Param	Iters	IS ↑	FID ↓	sFID ↓	Prec. ↑	Recall ↑	Sampling it/s ↑
DiT-XL/2 [40]	28	675 M	7,000 K	278.24	2.27	4.60	0.83	0.57	6.91
DiT-XL/2 [40]	28	675 M	2,000 K	240.22	2.73	4.46	0.83	0.55	6.91
DiT-XL/2 [40]	28	675 M	1,000 K	157.83	5.53	4.60	0.80	0.53	6.91
U-ViT-H/2 [1]	29	501 M	500 K	265.30	2.30	5.60	0.82	0.58	8.21
ShortGPT [36]	28⇒19	459 M	100 K	132.79	7.93	5.25	0.76	0.53	10.07
TinyDiT-D19 (KD)	28⇒19	459 M	100 K	242.29	2.90	4.63	0.84	0.54	10.07
TinyDiT-D19 (KD)	28⇒19	459 M	500 K	251.02	2.55	4.57	0.83	0.55	10.07
DiT-L/2 [40]	24	458 M	1,000 K	196.26	3.73	4.62	0.82	0.54	9.73
U-ViT-L [1]	21	287 M	300 K	221.29	3.44	6.58	0.83	0.52	13.48
U-DiT-L [50]	22	204 M	400 K	246.03	3.37	4.49	0.86	0.50	-
Diff-Pruning-50% [12]	28	338 M	100 K	186.02	3.85	4.92	0.82	0.54	10.43
Diff-Pruning-75% [12]	28	169 M	100 K	83.78	14.58	6.28	0.72	0.53	13.59
ShortGPT [36]	28⇒14	340 M	100 K	66.10	22.28	6.20	0.63	0.56	13.54
Flux-Lite [6]	28⇒14	340 M	100 K	54.54	25.92	5.98	0.62	0.55	13.54
Sensitivity Analysis [18]	28⇒14	340 M	100 K	70.36	21.15	6.22	0.63	**0.57**	13.54
Oracle (BK-SDM) [23]	28⇒14	340 M	100 K	141.18	7.43	6.09	0.75	0.55	13.54
TinyDiT-D14	28⇒14	340 M	100 K	151.88	5.73	4.91	0.80	0.55	13.54
TinyDiT-D14	28⇒14	340 M	500 K	198.85	3.92	5.69	0.78	0.58	13.54
TinyDiT-D14 (KD)	28⇒14	340 M	100 K	207.27	3.73	5.04	0.81	0.54	13.54
TinyDiT-D14 (KD)	28⇒14	340 M	500 K	234.50	2.86	4.75	0.82	0.55	13.54
DiT-B/2 [40]	12	130 M	1,000 K	119.63	10.12	5.39	0.73	0.55	28.30
U-DiT-B [50]	22	-	400 K	85.15	16.64	6.33	0.64	0.63	-
TinyDiT-D7 (KD)	14⇒7	173 M	500 K	166.91	5.87	5.43	0.78	0.53	26.81

Table 1. Layer pruning results for pre-trained DiT-XL/2. We focus on two settings: fast training with 100K optimization steps and sufficient fine-tuning with 500K steps. Both fine-tuning and Masked Knowledge Distillation (a variant of KD, see Sec. 4.4) are used for recovery.

might not be competitive at the beginning, but may emerge as competitive candidates with sufficient fine-tuning. After training, we retain those local structures with the highest probability and discard the additional update $\Delta\Phi$. Then, standard fine-tuning techniques can be applied for recovery.

4. Experiments

4.1. Experimental Settings

Our experiments were mainly conducted on Diffusion Transformers [40] for class-conditional image generation on ImageNet 256 × 256 [8]. For evaluation, we follow [9, 40] and report the Fréchet inception distance (FID), Sliding Fréchet Inception Distance (sFID), Inception Scores (IS), Precision and Recall using the official reference images [9]. Additionally, we also extend our methods to other models, including MARs [29] and SiTs [34]. Experimental details can be found in the following sections and appendix.

4.2. Results on Diffusion Transformers

DiT. This work focuses on the compression of DiTs [40]. We consider two primary strategies as baselines: the first involves using manually crafted patterns to eliminate layers. For instance, BK-SDM [23] employs heuristic assumptions to determine the significance of specific layers, such as the initial or final layers. The second strategy is based on systematically designed criteria to evaluate layer importance, such as analyzing the similarity between block inputs and outputs to determine redundancy [6, 36]; this approach typically aims to minimize performance degradation after pruning. Table 1 presents representatives from both strategies, including ShortGPT [36], Flux-Lite [6], Diff-Pruning [12], Sensitivity Analysis [18] and BK-SDM [23], which serve as baselines for comparison. Additionally, we compared our method to other architectures, such as UViT [1], U-DiT [50], and DTR [39], which have demonstrated improved training efficiency over conventional DiTs.

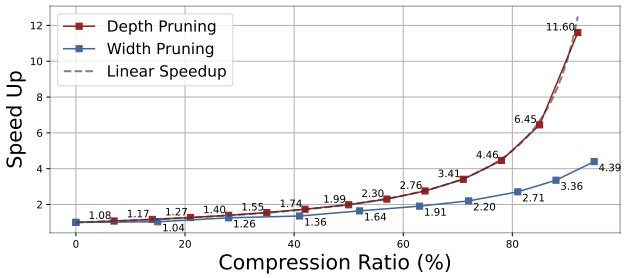


Figure 4. Depth pruning closely aligns with the theoretical linear speed-up relative to the compression ratio.

Method	Depth	Params	Epochs	FID	IS
MAR-Large	32	479 M	400	1.78	296.0
MAR-Base	24	208 M	400	2.31	281.7
TinyMAR-D16	32⇒16	277 M	40	2.28	283.4
SiT-XL/2	28	675 M	1,400	2.06	277.5
TinySiT-D14	28⇒14	340 M	100	3.02	220.1

Table 2. Depth pruning results on MARs [29] and SiTs [34].

Table 1 presents our findings on compressing a pre-trained DiT-XL/2 [40]. This model contains 28 transformer layers structured with alternating Attention and MLP layers. The proposed method seeks to identify shallow transformers with $\{7, 14, 19\}$ sub-layers from these 28 layers, to maximize the post-fine-tuning performance. With only 7% of the original training cost (500K steps compared to 7M steps), TinyDiT achieves competitive performance relative to both pruning-based methods and novel architectures. For instance, a DiT-L model trained from scratch for 1M steps achieves an FID score of 3.73 with 458M parameters. In contrast, the compressed TinyDiT-D14 model, with only 340M parameters and a faster sampling speed (13.54 it/s vs. 9.73 it/s), yields a significantly improved FID of 2.86. On parallel devices like GPUs, the primary bottleneck in transformers arises from sequential operations within each layer, which becomes more pronounced as the number of layers increases. Depth pruning mitigates this bottleneck by removing entire transformer layers, thereby reducing computational depth and optimizing the workload. By comparison, width pruning only reduces the number of neurons within each layer, limiting its speed-up potential. As shown in Figure 4, depth pruning closely matches the theoretical linear speed-up as the compression ratio increases, outperforming width pruning methods such as Diff-Pruning [12].

MAR & SiT. Masked Autoregressive (MAR) [29] models employ a diffusion loss-based autoregressive framework in a continuous-valued space, achieving high-quality image generation without the need for discrete tokenization. The MAR-Large model, with 32 transformer blocks, serves as the baseline for comparison. Applying our pruning method, we reduced MAR to a 16-block variant, TinyMAR-D16, achieving an FID of 2.28 and surpassing the performance of the 24-block MAR-Base model with only 10% of the original training cost (40 epochs vs. 400 epochs). Our approach also generalizes to Scalable Interpolant Transformers (SiT) [34], an extension of the DiT architecture that employs a flow-based interpolant framework to bridge data and noise distributions. The SiT-XL/2 model, comprising 28 transformer blocks, was pruned by 50%, creating the TinySiT-D14 model. This pruned model retains competitive performance at only 7% of the original training cost (100 epochs vs. 1400 epochs). As shown in Table 2, these results demonstrate that our pruning method is adaptable across different diffusion transformer variants, effectively reducing the model size and training time while maintaining strong performance.

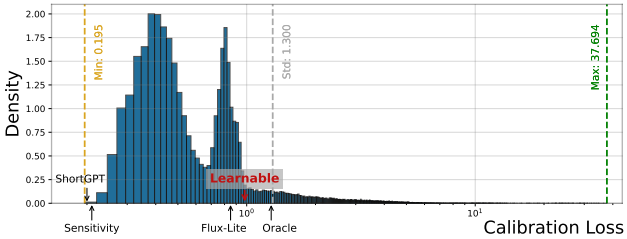


Figure 5. Distribution of calibration loss through random sampling of candidate models. The proposed learnable method achieves the best post-fine-tuning FID yet has a relatively high initial loss compared to other baselines.

Strategy	Loss	IS	FID	Prec.	Recall
Max. Loss	37.69	NaN	NaN	NaN	NaN
Med. Loss	0.99	149.51	6.45	0.78	0.53
Min. Loss	**0.20**	73.10	20.69	0.63	0.58
Sensitivity	0.21	70.36	21.15	0.63	**0.57**
ShortGPT [36]	0.20	66.10	22.28	0.63	0.56
Flux-Lite [6]	0.85	54.54	25.92	0.62	0.55
Oracle (BK-SDM)	1.28	141.18	7.43	0.75	0.55
Learnable	0.98	**151.88**	**5.73**	**0.80**	0.55

Table 3. Directly minimizing the calibration loss may lead to non-optimal solutions. All pruned models are fine-tuned *without* knowledge distillation (KD) for 100K steps. We evaluate the following baselines: (1) Loss – We randomly prune a DiT-XL model to generate 100,000 models and select models with different calibration losses for fine-tuning; (2) Metric-based Methods – such as Sensitivity Analysis and ShortGPT; (3) Oracle – We retain the first and last layers while uniformly pruning the intermediate layers following [23]; (4) Learnable – The proposed learnable method.

4.3. Analytical Experiments

Is Calibration Loss the Primary Determinant? An essential question in depth pruning is how to identify redundant layers in pre-trained diffusion transformers. A common approach involves minimizing the calibration loss, based on the assumption that a model with lower calibration loss after pruning will exhibit superior performance. However, we demonstrate in this section that this hypothesis may not hold for diffusion transformers. We begin by examining the solution space through random depth pruning at a 50% ratio, generating 100,000 candidate models with calibration losses ranging from 0.195 to 37.694 (see Figure 5). From these candidates, we select models with the highest and lowest calibration losses for fine-tuning. Notably, both models result in unfavorable outcomes, such as

Pattern	ΔW	IS ↑	FID ↓	sFID ↓	Prec. ↑	Recall ↑
1:2	LoRA	54.75	33.39	29.56	0.56	0.62
2:4	LoRA	53.07	34.21	27.61	0.55	0.63
7:14	LoRA	34.97	49.41	28.48	0.46	0.56
1:2	Full	53.11	35.77	32.68	0.54	0.61
2:4	Full	53.63	34.41	29.93	0.55	0.62
7:14	Full	45.03	38.76	31.31	0.52	0.62
1:2	Frozen	45.08	39.56	31.13	0.52	0.60
2:4	Frozen	48.09	37.82	31.91	0.53	0.62
7:14	Frozen	34.09	49.75	31.06	0.46	0.56

Table 4. Performance comparison of TinyDiT-D14 models compressed using various pruning schemes and recoverability estimation strategies. All models are fine-tuned for 10,000 steps, and FID scores are computed on 10,000 sampled images with 64 timesteps.

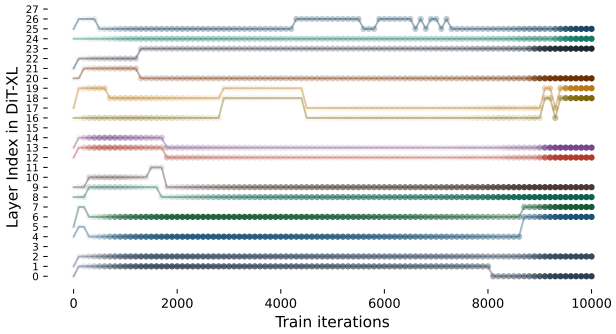


Figure 6. Visualization of the 2:4 decisions in the learnable pruning, with the confidence level of each decision highlighted through varying degrees of transparency. More visualization results for 1:2 and 7:14 schemes are available in the appendix.

unstable training (NaN) or suboptimal FID scores (20.69), as shown in Table 3. Additionally, we conduct a sensitivity analysis [18], a commonly used technique to identify crucial layers by measuring loss disturbance upon layer removal, which produces a model with a low calibration loss of 0.21. However, this model's FID score is similar to that of the model with the lowest calibration loss. Approaches like ShortGPT [36] and a recent approach for compressing the Flux model [6], which estimate similarity or minimize mean squared error (MSE) between input and output states, reveal a similar trend. In contrast, methods with moderate calibration losses, such as Oracle (often considered less competitive) and one of the randomly pruned models, achieve FID scores of 7.43 and 6.45, respectively, demonstrating significantly better performance than models with minimal calibration loss. These findings suggest that, while calibration loss may influence post-fine-tuning performance to some extent, it is not the primary determinant for diffusion transformers. Instead, the model's capacity for performance recovery during fine-tuning, termed "recoverability," appears to be more critical. Notably, assessing recoverability using traditional metrics is challenging, as it requires a learning process across the entire dataset. This observation also explains why the proposed method achieves superior results (5.73) compared to baseline methods.

Learnable Modeling of Recoverability. To overcome the limitations of traditional metric-based methods, this study introduces a learnable approach to jointly optimize pruning and model recoverability. Table 3 illustrates different configurations of the learnable method, including the local pruning scheme and update strategies for recoverability estimation. For a 28-layer DiT-XL/2 with a fixed 50% layer pruning rate, we examine three splitting schemes: 1:2, 2:4, and 7:14. In the 1:2 scheme, for example, every two transformer layers form a local block, with one layer pruned. Larger blocks introduce greater diversity but significantly expand the search space. For instance, the 7:14 scheme divides the model into two segments, each retaining 7 layers, resulting in $\binom{14}{7} \times 2 = 6{,}864$ possible solutions. Conversely, smaller blocks significantly reduce optimization difficulty and offer greater flexibility. When the distribution of one block converges, the learning on other blocks can still progress. As shown in Table 3, the 1:2 configuration achieves the optimal performance after 10K fine-tuning iterations. Additionally, our empirical findings underscore the effectiveness of recoverability estimation using LoRA or full fine-tuning. Both methods yield positive post-fine-tuning outcomes, with LoRA achieving superior results (FID = 33.39) compared to full fine-tuning (FID = 35.77) under the 1:2 scheme, as LoRA has fewer trainable parameters (0.9% relative to full parameter training) and can adapt more efficiently to the randomness of sampling.

Visualization of Learnable Decisions. To gain deeper insights into the role of the learnable method in pruning, we visualize the learning process in Figure 6. From bottom to top, the i-th curve represents the i-th layer of the pruned model, displaying its layer index in the original DiT-XL/2. This visualization illustrates the dynamics of pruning decisions over training iterations, where the transparency of each data point indicates the probability of being sampled. The learnable method shows its capacity to explore and handle various layer combinations. Pruning decisions for certain layers, such as the 7-th and 8-th in the compressed model, are determined quickly and remain stable throughout the process. In contrast, other layers, like the 0-th layer, require additional fine-tuning to estimate their recoverability. Notably, some decisions may change in the later stages once these layers have been sufficiently optimized. The training process ultimately concludes with high sampling probabilities, suggesting a converged learning process with distributions approaching a one-hot configuration.

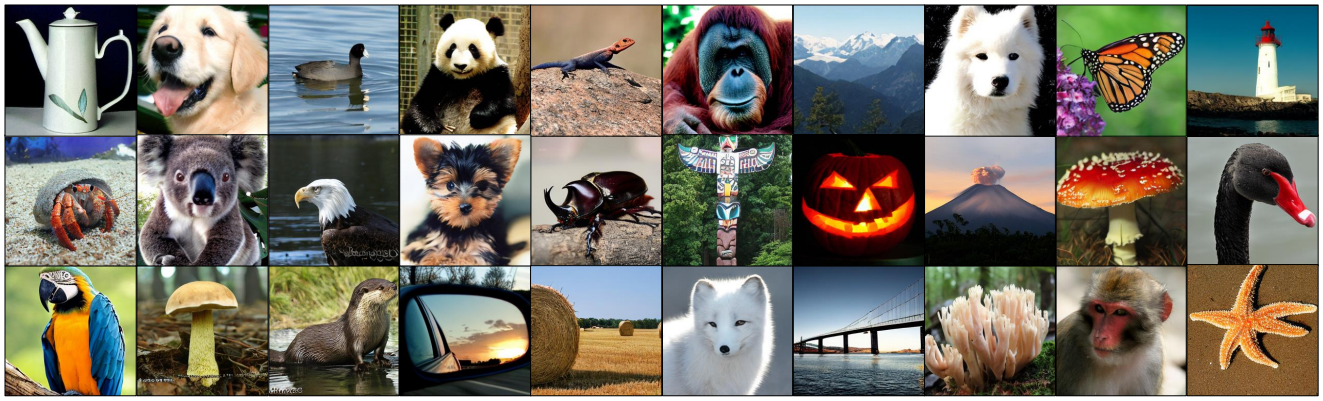
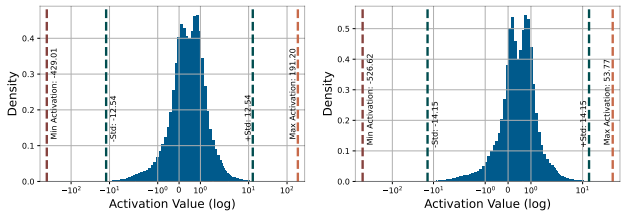


Figure 7. Images generated by TinyDiT-D14 on ImageNet 224×224, pruned and distilled from a DiT-XL/2.



(a) DiT-XL/2 (Teacher) (b) TinyDiT-D14 (Student)

Figure 8. Visualization of massive activations [47] in DiTs. Both teacher and student models display large activation values in their hidden states. Directly distilling these massive activations may result in excessively large losses and unstable training.

fine-tuning Strategy	Init. Distill. Loss	FID @ 100K
fine-tuning	-	5.79
Logits KD	-	4.66
RepKD	2840.1	NaN
Masked KD (0.1σ)	15.4	NaN
Masked KD (2σ)	387.1	3.73
Masked KD (4σ)	391.4	3.75

Table 5. Evaluation of different fine-tuning strategies for recovery. Masked RepKD ignores those massive activations ($|x| > k\sigma_x$) in both teacher and student to enables effective knowledge transfer.

4.4. Knowledge Distillation for Recovery

In this work, we also explore Knowledge Distillation (KD) as an enhanced fine-tuning method. As demonstrated in Table 5, we first apply the vanilla knowledge distillation approach proposed by Hinton [20] to fine-tune the TinyDiT-D14, supervised by the outputs of the DiT-XL/2, which effectively reduces the FID at 100K steps from 5.79 to 4.66.

Masked Knowledge Distillation. Further, we evaluate representation distillation (RepKD) [23, 42] to transfer hidden states from the teacher to the student. It is important to note that depth pruning does not alter the hidden dimension of diffusion transformers, allowing for direct alignment of intermediate hidden states. For practical implementation, we use the block defined in Section 3.2 as the basic unit, ensuring that the pruned local structure in the pruned DiT aligns with the output of the original structure in the teacher model. However, we encountered significant training difficulties with this straightforward RepKD approach due to massive activations in the hidden states, where both teacher and student models occasionally exhibit large activation values, as shown in Figure 8. Directly distilling these extreme activations can result in excessively high loss values, impairing the performance of the student model. This issue has also been observed in other transformer-based generative models, such as certain LLMs [47]. To address this, we propose a Masked RepKD variant that selectively excludes these massive activations during knowledge transfer. We employ a simple thresholding method, $|x - \mu_x| < k\sigma_x$, which ignores the loss associated with these extreme activations. As shown in Table 5, the Masked RepKD approach with moderate thresholds of 2σ and 4σ achieves satisfactory results, demonstrating the robustness of our method.

Visualization of Generated Images. In Figure 7, We visualize the generated images of the learned TinyDiT-D14, distilled from an off-the-shelf DiT-XL/2 model. More visualization results for SiTs and MARs can be found in the supplementary materials.

5. Conclusions

This work introduces TinyFusion, a learnable method for accelerating diffusion transformers by removing redundant layers. It models the recoverability of pruned models as an optimizable objective and incorporates differentiable sampling for end-to-end training. Our method generalizes to various architectures like DiTs, MARs and SiTs.

Acknowledgement

This project is supported by the National Research Foundation, Singapore, under its AI Singapore Programme (AISG Award No: AISG2-RP-2021-023), and the Singapore Ministry of Education Academic Research Fund Tier 1 (WBS: A-0009440-01-00).

References

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

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[3] Thibault Castells, Hyoung-Kyu Song, Bo-Kyeong Kim, and Shinkook Choi. Ld-pruner: Efficient pruning of latent diffusion models using task-agnostic insights. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 821–830, 2024.

[4] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Conference on Computer Vision and Pattern Recognition*, pages 11315–11325, 2022.

[5] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis, 2023.

[6] Javier Martín Daniel Verdú. Flux.1 lite: Distilling flux1.dev for efficient text-to-image generation. 2024.

[7] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

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

[10] Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. *arXiv preprint arXiv:1910.10073*, 2019.

[11] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.

[12] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

[13] Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. Maskllm: Learnable semi-structured sparsity for large language models. *arXiv preprint arXiv:2409.17481*, 2024.

[14] Zhengcong Fei, Mingyuan Fan, Changqian Yu, Debang Li, and Junshi Huang. Scaling diffusion transformers to 16 billion parameters. *arXiv preprint arXiv:2407.11633*, 2024.

[15] Zhengcong Fei, Mingyuan Fan, Changqian Yu, Debang Li, Youqiang Zhang, and Junshi Huang. Dimba: Transformer-mamba diffusion models. *arXiv preprint arXiv:2406.01159*, 2024.

[16] Shanghua Gao, Zhijie Lin, Xingyu Xie, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Editanything: Empowering unparalleled flexibility in image editing and generation. In *Proceedings of the 31st ACM International Conference on Multimedia, Demo track*, 2023.

[17] Emil Julius Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*. US Government Printing Office, 1954.

[18] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[19] Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Ptqd: Accurate post-training quantization for diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

[20] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[21] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[22] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[23] Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. Bk-sdm: Architecturally compressed stable diffusion for efficient text-to-image generation. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*, 2023.

[24] Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. Shortened llama: A simple depth pruning for large language models. *arXiv preprint arXiv:2402.02834*, 11, 2024.

[25] PKU-Yuan Lab and Tuzhan AI etc. Open-sora-plan, 2024.

[26] Black Forest Labs. FLUX, 2024.

[27] Youngwan Lee, Yong-Ju Lee, and Sung Ju Hwang. Dit-pruner: Pruning diffusion transformer models for text-to-image synthesis using human preference scores.

[28] Youngwan Lee, Kwanyong Park, Yoorhim Cho, Yong-Ju Lee, and Sung Ju Hwang. Koala: self-attention matters in knowledge distillation of latent diffusion models for memory-efficient and fast image synthesis. *arXiv e-prints*, pages arXiv–2312, 2023.

[29] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *arXiv preprint arXiv:2406.11838*, 2024.

[30] Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-diffusion: Quantizing diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17535–17545, 2023.

[31] Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. *Advances in Neural Information Processing Systems*, 36, 2024.

[32] Shanchuan Lin, Anran Wang, and Xiao Yang. Sdxl-lightning: Progressive adversarial diffusion distillation. *arXiv preprint arXiv:2402.13929*, 2024.

[33] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

[34] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. *arXiv preprint arXiv:2401.08740*, 2024.

[35] Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching, 2024.

[36] Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*, 2024.

[37] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

[38] Zanlin Ni, Yulin Wang, Renping Zhou, Jiayi Guo, Jinyi Hu, Zhiyuan Liu, Shiji Song, Yuan Yao, and Gao Huang. Revisiting non-autoregressive transformers for efficient image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7007–7016, 2024.

[39] Byeongjun Park, Sangmin Woo, Hyojun Go, Jin-Young Kim, and Changick Kim. Denoising task routing for diffusion models. *arXiv preprint arXiv:2310.07138*, 2023.

[40] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

[41] David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

[42] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[43] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[44] Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1972–1981, 2023.

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

[46] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

[47] Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.

[48] Yao Teng, Yue Wu, Han Shi, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Dim: Diffusion mamba for efficient high-resolution image synthesis. *arXiv preprint arXiv:2405.14224*, 2024.

[49] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. 2024.

[50] Yuchuan Tian, Zhijun Tu, Hanting Chen, Jie Hu, Chao Xu, and Yunhe Wang. U-dits: Downsample tokens in u-shaped diffusion transformers. *arXiv preprint arXiv:2405.02730*, 2024.

[51] Kafeng Wang, Jianfei Chen, He Li, Zhenpeng Mi, and Jun Zhu. Sparsedm: Toward sparse efficient diffusion models. *arXiv preprint arXiv:2404.10445*, 2024.

[52] Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Yujun Lin, Zhekai Zhang, Muyang Li, Yao Lu, and Song Han. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. *arXiv preprint arXiv:2410.10629*, 2024.

[53] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4): 1–39, 2023.

[54] Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. In *Conference on Artificial Intelligence (AAAI)*, 2022.

[55] Tao Yu, Runseng Feng, Ruoyu Feng, Jinming Liu, Xin Jin, Wenjun Zeng, and Zhibo Chen. Inpaint anything: Segment anything meets image inpainting. *arXiv preprint arXiv:2304.06790*, 2023.

[56] Dingkun Zhang, Sijia Li, Chen Chen, Qingsong Xie, and Haonan Lu. Laptop-diff: Layer pruning and normalized distillation for compressing diffusion models. *arXiv preprint arXiv:2404.11098*, 2024.

[57] Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. *arXiv preprint arXiv:2408.12588*, 2024.

[58] Yang Zhao, Yanwu Xu, Zhisheng Xiao, and Tingbo Hou. Mobilediffusion: Subsecond text-to-image generation on mobile devices. *arXiv preprint arXiv:2311.16567*, 2023.

[59] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all, 2024.