SparseDiT: Token Sparsification for Efficient Diffusion Transformer

Shuning Chang[1 2 3]   **Pichao Wang**[2*]   **Jiasheng Tang**[2 3]   **Fan Wang**[2 3]   **Yi Yang**[1]

[1]Zhejiang University   [2]Damo Academy, Alibaba Group   [3]Hupan Lab

`shuning.csn@alibaba-inc.com`

Abstract

Diffusion Transformers (DiT) are renowned for their impressive generative performance; however, they are significantly constrained by considerable computational costs due to the quadratic complexity in self-attention and the extensive sampling steps required. While advancements have been made in expediting the sampling process, the underlying architectural inefficiencies within DiT remain underexplored. We introduce SparseDiT, a novel framework that implements token sparsification across spatial and temporal dimensions to enhance computational efficiency while preserving generative quality. Spatially, SparseDiT employs a tri-segment architecture that allocates token density based on feature requirements at each layer: Poolingformer in the bottom layers for efficient global feature extraction, Sparse-Dense Token Modules (SDTM) in the middle layers to balance global context with local detail, and dense tokens in the top layers to refine high-frequency details. Temporally, SparseDiT dynamically modulates token density across denoising stages, progressively increasing token count as finer details emerge in later timesteps. This synergy between SparseDiT's spatially adaptive architecture and its temporal pruning strategy enables a unified framework that balances efficiency and fidelity throughout the generation process. Our experiments demonstrate SparseDiT's effectiveness, achieving a 55% reduction in FLOPs and a 175% improvement in inference speed on DiT-XL with similar FID score on 512×512 ImageNet, a 56% reduction in FLOPs across video generation datasets, and a 69% improvement in inference speed on PixArt-α on text-to-image generation task with a 0.24 FID score decrease. SparseDiT provides a scalable solution for high-quality diffusion-based generation compatible with sampling optimization techniques. Code is available at `https://github.com/changsn/SparseDiT`.

1   Introduction

Diffusion models [18, 12, 43, 3] have emerged as powerful tools in visual generation, producing photorealistic images and videos by gradually refining structured content from noise. Leveraging the scalability of Transformers, Diffusion Transformer (DiT) models [41, 2] extend these capabilities to more complex and detailed tasks [13, 62] and form the backbone of advanced generation frameworks such as Sora [6]. Yet, despite their impressive generative capabilities, DiT models face significant computational limitations that restrict their broader applicability, especially in scenarios where efficiency is paramount.

The computational burden of DiT models arises mainly from the need for numerous sampling steps in the denoising process and the quadratic complexity of Transformer structures with respect to the number of tokens. Although recent research has focused on reducing this burden by accelerating the sampling process through techniques such as ODE solvers [51, 30], flow-based methods [26, 28],

---

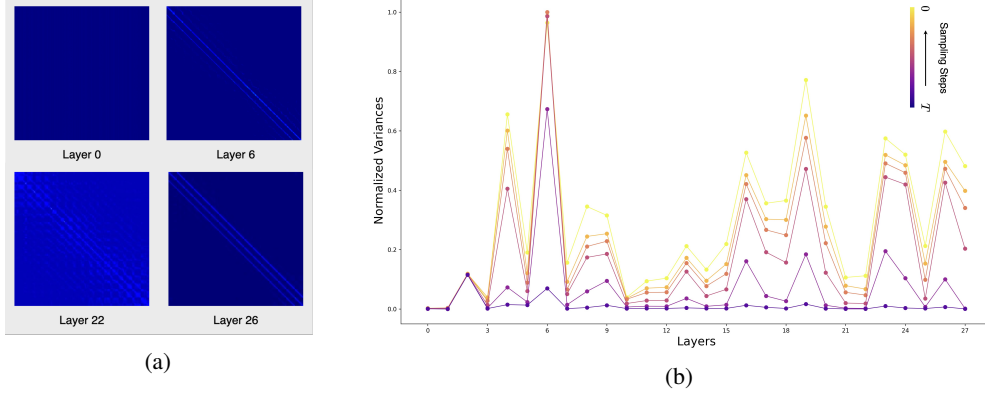[*]Work done at Alibaba Group, and now affiliated with Amazon.

Figure 1: (a) The attention maps in different self-attention layers. Zoom-in for better visibility. (b) The normalized variances of attention maps in DiT-XL. Different curve lines represent different sampling steps.

and knowledge distillation [53, 47, 32, 35, 63], these approaches often overlook the core architectural inefficiencies of DiT itself. Unlike U-Net-based architectures [44], which mitigate complexity through a contracting-expansive structure, DiT's reliance on token-level self-attention incurs high costs that scale with model size and token count. This bottleneck highlights a critical need for DiT-specific innovations that manage token density intelligently, balancing efficiency and quality without sacrificing generation fidelity.

To address this challenge, we examine the distribution of attention across DiT's layers and sampling steps, seeking to uncover structural patterns that could guide more efficient token management. As illustrated in Figure 1a, attention maps reveal that tokens capture different levels of granularity across layers: in the bottom layers, such as layer 0, attention maps approximately appear uniform distribution, indicating a focus on broad, global features akin to global pooling. Meanwhile, middle layers alternate in their attention, with certain layers (*e.g.*, layer 6 and layer 26) capturing local details, while others (*e.g.*, layer 22) emphasize global structure. This observation is quantified in Figure 1b, where the variance in attention scores highlights DiT's consistent pattern of alternation between global and local feature extraction across layers. In addition to spatial analysis, Figure 1b also reveals insights across sampling steps. We observe that, while the pattern of alternation between global and local focus remains stable, attention variance increases as the denoising process advances, indicating a growing emphasis on local information at lower-noise stages. This analysis yields three core insights: (1) the initial self-attention layers exhibit minimal variance, showing low discrepancy in feature extraction across tokens; (2) DiT's architecture inherently alternates between global and local feature focus across layers, a pattern consistent across all sampling steps; and (3) as denoising progresses, the model increasingly prioritizes local details, dynamically adapting to heightened demands for detail at later stages.

Building on these insights, we propose SparseDiT, which reimagines token density management as a dynamic process adapting across both spatial layers and temporal stages. Spatially, shallow layers capture smooth, global features, making complex self-attention less efficient. Subsequent layers alternate between high-frequency local details and low-frequency global information, where sparse tokens efficiently capture global features, and dense tokens refine local details. Temporally, as denoising advances, the need for local detail increases, motivating a dynamic token adjustment strategy. This dual-layered adaptation integrates layer-wise token modulation—balancing efficiency and detail within each sampling step—and a timestep-wise pruning strategy that adjusts token density dynamically. Together, these spatial and temporal adaptations allow SparseDiT to achieve computational efficiency without sacrificing high-fidelity detail, as reflected in SparseDiT's architectural design and pruning strategy, tailored to meet the unique spatial and temporal demands of the generation process.

SparseDiT Architecture: SparseDiT's architecture employs a three-segment design that aligns token density with the features each layer captures. In the bottom layers, we replace self-attention with a poolingformer structure to capture broad global features through average pooling, reducing computation while preserving essential structure. In the middle layers, SparseDiT introduces Sparse-Dense Token Modules (SDTM), blending sparse tokens for global structure with dense tokens to refine local details. This design enables SparseDiT to retain efficiency while maintaining stability and

(a) Architecture of SparseDiT.
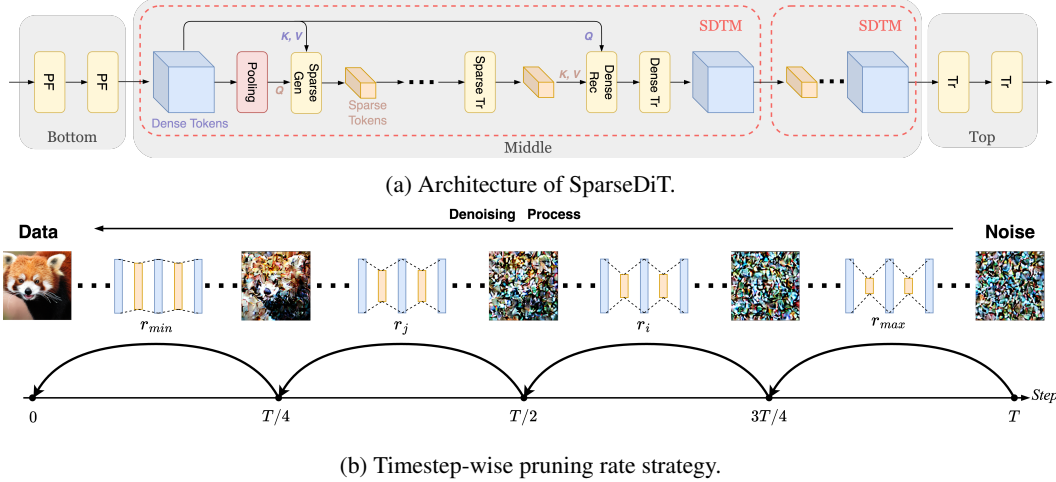

(b) Timestep-wise pruning rate strategy.

Figure 2: Architecture of SparseDiT and timestep-wise pruning rate strategy. The architecture consists of three segments: bottom, middle, and top. The bottom segment includes poolingformers (PF). The middle segment comprises multiple sparse-dense token modules (SDTMs), where sparse token generation transformers ("Sparse Gen") and dense token recovery transformers ("Dense Rec") alternate to balance global and detailed information processing. The top segment contains standard transformers ("Tr") for final processing. During the denoising process, we apply varying pruning rates r at different stages (depicted as four stages).

detail. In the top layers, the model processes all tokens densely, focusing on high-frequency details for refined output quality.

Timestep-Wise Pruning Rate Strategy: Complementing spatial adjustments, SparseDiT's timestep-wise pruning rate strategy adapts token density across denoising stages. In early stages, where broad, low-frequency structures dominate, SparseDiT applies a high pruning rate, conserving resources. As the process progresses and intricate details emerge, the pruning rate decreases, allowing for a gradual increase in token density. This adjustment aligns computational effort with each stage's complexity, dynamically balancing efficiency and detail to optimize resources for high-fidelity output.

Our empirical results substantiate the effectiveness of SparseDiT's integrated approach. SparseDiT reduces the FLOPs of DiT-XL by 55% and improves inference speed by 175% on 512×512 ImageNet [11] images, with only a increase of 0.09 in FID score [17]. On the Latte-XL dataset [34], SparseDiT achieves a 56% reduction in FLOPs across standard video generation datasets, including FaceForensics [45], SkyTimelapse [60], UCF101 [54], and Taichi-HD [50]. Additionally, on the more challenging text-to-image generation task, we achieve a 69% improvement in inference speed on PixArt-α with a 0.24 FID score reduction. These results demonstrate SparseDiT's capability to provide an efficient, high-quality architecture for diffusion-based generation, compatible with further sampling optimization techniques for enhanced efficiency.

2 Related works

Efficient sampling process. This direction focuses on optimizing sampling steps in diffusion models. Some approaches [51, 30, 1, 31] design new solvers to achieve faster sampling with fewer steps. Consistency models (CMs) [53, 52, 32, 58, 21] are closely related to diffusion models, achieving by distilling pre-trained diffusion models or training from scratch. These models learn a one-step mapping between noise and data, and all the points of the sampling trajectory map to the same initial data point. Flow-based [26, 28, 72, 13, 71] approaches straighten the transport trajectories among different distributions and thereby realize efficient inference. Knowledge distillation are applied by some methods [53, 47, 32, 35, 63] to reduce the sampling steps and match the output of the combined conditional and unconditional models. Parallel sampling techniques [49, 70, 56] employ Picard-Lindelöf iteration or Fourier neural operators for solving SDE/ODE.

Efficient model structure. Another avenue focuses on improving inference time within a single model evaluation. Some methods attempt to compress the size of diffusion models via pruning [14]

| (a) Standard DiT-XL. | (b) Modifying first two attention in DiT-XL. |

Figure 3: Comparison of images generated by the standard DiT-XL model and a modified DiT-XL model where the first two attention maps are replaced with full-one matrices without fine-tuning. Both sets of images exhibit similar visual quality.

or quantization [25, 48]. Spectral Diffusion [61] boosts structure design by incorporating frequency dynamics and priors. OMS-DPM [27] proposes a model schedule that selects small models or large models at specific sampling steps to balance generation quality and inference speed. Deepcache [33], TokenCache [29], and DiTFastAttn[66] reuse or share similar features across sampling steps. The early stopping mechanism in diffusion is explored in [24, 37, 55].

Most of the above methods are designed for general diffusion models. Recently, the Diffusion Transformer (DiT) has emerged as a more potential backbone, surpassing the previously dominant U-Net architectures. Unlike U-Net-based models, DiT's inefficiency stems largely from the self-attention mechanism. Limited research has specifically addressed the efficiency of DiT-based models. Pu et al. [42] identifies the redundancies in the self-attention and adopts timestep dynamic mediator tokens to compress attention complexity, achieving some performances but limited FLOPs reduction, as this approach primarily reduces the number of keys and values rather than the overall tokens count. In Vision transformers for classification task, numerous methods [59, 9, 15, 7, 4, 73, 36, 69, 67] successfully removed redundant tokens to improve performance-efficiency trade-offs. ToMeSD [5] is the first to explore token reduction in diffusion models. However, its performance on DiT was suboptimal, as shown in [37, 68]. U-Net can be regarded as a sparse network due to its contracting and expansive structure. EDT [10] applies a novel architecture similar to U-Net, while they still have 1.4 FID score gap compared to its baseline model MDTv2 [16] via 2,000K iterations. These findings indicate that directly transferring token reduction techniques from classification tasks or reusing U-Net structures may be insufficient for DiT. Instead, a specialized token reduction strategy is needed. DyDiT [68] and DiffCR [64] design dynamic networks to compress DiT from multiple dimensions, such as token, layer, attention head, channel.

3 Method

We first introduce SparseDiT architecture in Section 3.1, followed by our timestep-wise pruning rate strategy in Section 3.2. Finally, Section 3.3 details the initialization and fine-tuning of our network.

3.1 SparseDiT

Overview architecture. The SparseDiT architecture is illustrated in Figure 2a, where the transformer layers of DiT based model are divided into three groups: bottom, middle, and top. The bottom layers leverage poolingformers to efficiently capture global features. The middle layers contain multiple sparse-dense token modules (SDTM), which decouple the representation process into global structure capture and local details enhancement using sparse and dense tokens, respectively. The top layers retain the standard transformers, processing dense tokens to generate the final predictions at each sampling step. The primary computational savings are achieved through sparse tokens, hence the majority of transformer layers are located in the middle section to maximize efficiency.

Poolingformer in the bottom layers. As shown in the first subfigure in Figure 1a, attention scores in the bottom layers exhibit a nearly uniform distribution, with each token evenly extracting global features, akin to a global average pooling operator. To investigate further, we conducted a toy experiment where the attention maps in the first two transformer layers were replaced with

matrices filled with a constant value (*e.g.*, 1), without any fine-tuning. Figure 3 shows that, using the same initial noise and random seed, the images generated by the original DiT-XL and the modified DiT-XL are nearly identical, suggesting that complex self-attention calculations offer limited additional information. Given that self-attention in the bottom layers captures global features, we further question whether sparse tokens can be used in these layers. However, experiments (in Table 6) demonstrated that applying sparse tokens in the bottom layers results in unstable training, highlighting the necessity of retaining complete tokens in these layers.

Based on the above analysis, we adopt poolingformers to replace the original transformers. In poolingformer, we remove queries and keys as attention maps are not computed. Instead, we perform a global average pooling on the value $V \in \mathbb{R}^{N \times C}$ and integrate it into input tokens X, which can be represented as

$$X = X + \bar{V}, \tag{1}$$

where $\bar{V}$ is the mean along the dimension of N, *i.e.*, $\bar{v} = \frac{1}{N} \sum V (\bar{v} \in \mathbb{R}^{1 \times C})$, and then repeating N times to shape $N \times C$. The parameters from adaLN block are omitted in all the equations for brevity. Our poolingformer can be viewed as a special case of the model in [65], behaving identically when the pooling kernel size is equal to the input size.

Sparse-dense token module. We present the sparse-dense token module (SDTM) to generate sparse and dense tokens processed by the corresponding transformer layers. The high-level idea is to decouple global structure extraction and local detail extraction. Sparse tokens capture global structure information and reduce computational cost, while dense tokens enhance local detail and stabilize training. Sparse and dense tokens are converted to each other within SDTM.

Initially, SDTM introduces a set of sparse tokens $X_s \in \mathbb{R}^{M \times C}$, where M is the number of sparse tokens, typically $M \ll N$. These sparse tokens can be initialized by adaptively pooling the dense tokens X. We define pruning rate $r = 1 - M/N$ to represent the sparsity degree. To store the structure information, we first reshape the dense tokens $X \in \mathbb{R}^{N \times C}$ into the latent shape. For instance, if the input is an image, we reshape it into shape $H \times W \times C$, then pool it across the spatial dimensions to shape $H' \times W' \times C$, where $H' \times W' = M$. The intentions of spatial pooling initialization are two-fold. First, the initial sparse tokens can be distributed uniformly in space and the representation of each sparse token is associated with a specific spatial location, which is beneficial for downstream tasks such as image editing [20]. Second, it can prevent the semantic tokens from collapsing to one point in the following layers. Sparse token interact with full-size dense tokens via an attention layer to integrate the global information:

$$X_s = X_s + MHA(X_s, X, X), \tag{2}$$

where the triplet input of MHA are queries, keys, and values in turn.

The generated sparse tokens are fed into subsequent transformer layers, referred to as sparse transformers. Since $M \ll N$, our SDTM can substantially reduce computational cost.

Following this, we restore dense tokens from sparse tokens. First, we reshape the sparse tokens $X_s \in \mathbb{R}^{M \times C}$ into structure shape $H' \times W' \times C$ and upsample it to the same shape as input dense tokens $X \in \mathbb{R}^{H \times W \times C}$. We introduce two linear layers to combine upsampling sparse tokens with dense tokens, which is represented as:

$$X_{merged} = UpSample(X_s) \cdot W_1 + X \cdot W_2, \tag{3}$$

where $W_1, W_2 \in \mathbb{R}^{C \times C}$ are the weights of two linear layers. Then, to further incorporate sparse tokens, we utilize an attention layer to perform a reverse operation of the generation of sparse token to produce the restored dense tokens, which is written as:

$$X = X_{merged} + MHA(X_{merged}, X_s, X_s). \tag{4}$$

At the end of SDTM, several transformer layers, termed dense transformers, process full-size dense tokens to enhance local details.

We cascade multiple SDTMs in our network. By repeating sparse and dense tokens, the network effectively preserves both structural and detailed information, achieving substantial reductions in computational cost while maintaining high-quality generation outputs.

Table 1: Comparison of performance between SparseDiT and DiT for class-conditional image generation task on ImageNet.

Resolution	Model	FLOPs (G)	Throughput (img/s)	FID ↓	sFID ↓	IS ↑	Precision ↑	Recall ↑
256×256	DiT-B	23.01	7.84	9.07	5.44	121.07	0.74	0.54
	Ours ($r \in [0.61, 0.86]$)	14.34 (-38%)	13.2 (+68%)	8.23	6.20	134.30	0.74	0.54
256×256	DiT-XL	118.64	1.58	2.27	4.60	278.24	0.83	0.57
	Ours ($r \in [0.44, 0.61]$)	88.91 (-25%)	2.13 (+35%)	2.23	4.62	278.91	0.84	0.58
	Ours ($r \in [0.61, 0.86]$)	68.05 (-43%)	2.95 (+87%)	2.38	4.82	276.39	0.82	0.58
512×512	DiT-XL	525	0.249	3.04	5.02	240.82	0.84	0.54
	Ours ($r \in [0.61, 0.86]$)	286 (-46%)	0.609 (+145%)	2.96	5.00	242.4	0.84	0.54
	Ours ($r \in [0.90, 0.96]$)	235 (-55%)	0.685 (+175%)	3.13	5.41	236.56	0.83	0.52

(a) Images generated by DiT.

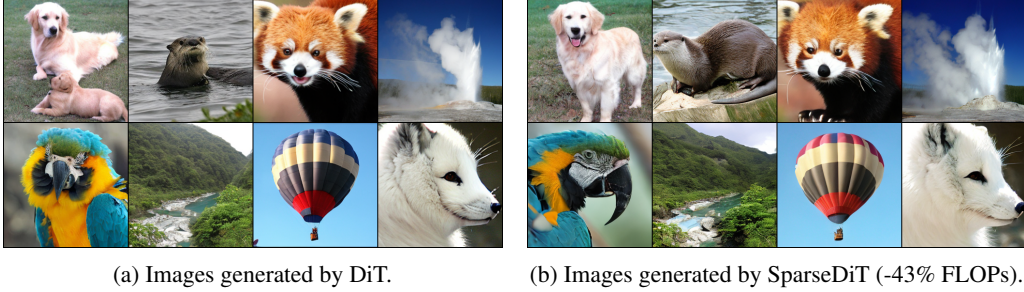(b) Images generated by SparseDiT (-43% FLOPs).

Figure 4: Generating images from DiT-XL and SparseDiT-XL with the same random seed.

3.2 Timestep-wise pruning rate

We have observed that the tokens display varying denoising behavior over sampling steps. They generate the low-frequency global structure information in the early denoising stage and the high frequency details in the late denoising stage. The token count requirement is progressively increase along sampling steps. We exploit this by dynamically adjusting the pruning rate r across sampling steps, increasing tokens as sampling progresses.

Given T sampling steps, we propose a sample-specific approach to dynamically adjust the pruning rate r, thereby controlling the number of sparse tokens across the sampling steps. We define a range for r, such that $r \in [r_{\min}, r_{\max}]$. Observing that generation quality is highly relative with the later denoising stages (in Section 4.4), we hold r constant at $r_{\min}$ for the first $T/4$ sampling steps. Subsequently, we adjust the pruning rate r linearly based on the current sampling step t_i. The specific formula for r is provided as follows:

$$r = \begin{cases} r_{min}, & t_i < T/4, \\ \frac{4T-4}{3T} r_{min} + \frac{4-T}{3T} r_{max}, & T/4 \leq t_i < T. \end{cases} \tag{5}$$

During training, we sample step t_i and compute the corresponding r according to Eq. 5. However, the input tokens should be the same to train the model in batch, and the randomness of the sampling from T should also be maintained. To solve this contradiction, we modify the linear function of Eq. 5 when $T/4 \leq t_i < T$ to a piecewise function. Normally, the model is trained by multiple GPUs. We request t_i sampled in a specific piece in each GPU. Therefore, in each iteration, it can achieve both uniformly random sampling and batch training.

3.3 Initialization and fine-tuning

Our method fine-tunes pre-trained DiT models to improve efficiency. The processes of generating sparse tokens and restoring dense tokens utilize off-the-shelf transformers, avoiding the need for additional networks. During fine-tuning, the parameters of transformers in DiT-based model are loaded into the corresponding transformers in our SparseDiT, with two exceptions: (1) queries and keys are absent in Poolingformers, hence related parameters in pre-trained model are not required; (2) the weights W_1 and W_2 in Eq. 3 are initialized by full zeros and identity matrix, receptively.

6

Table 2: FVD scores of SparseDiT-XL in class-conditional video generation task on four mainstream video generation datasets. We do not find the official reference sets of these 4 dataset when we compute FVD scores, hence we build the reference sets by ourselves and use the official checkpoint to re-compute the FVD scores of Latte-XL in this table.

Model	FLOPs (G)	Throughput (clip/s)	FaceForensic	SkyTimelapse	UCF101	Taichi-HD
Latte-XL*	1894	0.108	24.10	40.78	284.54	89.63
Ours ($r \in [0.80, 0.93]$)	828 (-56%)	0.228 (+111%)	24.51	39.02	288.90	84.13

4 Experiments

SparseDiT is applied in three representative DiT-based models, DiT [41], Latte [34], and PixArt-α [8] for class-conditional image generation, class-conditional video generation, and text-to-image generation, respectively.

Fine-tuning overhead. All training settings and hyperparameters follow their respective papers. Fine-tuning requires approximately 6% of the time needed for training from scratch, *e.g.*, 400K iterations for DiT-XL fine-tuning.

4.1 Class-conditional image generation

Experimental setting. We conduct our experiments on ImageNet-1k [11] at resolutions of 256×256 and 512×512, following the protocol established in DiT. For DiT-XL, the model consists of 2, 24, and 2 transformers in the bottom, middle, and top segments, respectively. The middle segment includes 4 sparse-dense token modules, which comprise 1 sparse token generation transformer, 3 sparse transformers, 1 dense token recovery transformer, and 1 dense transformer. For DiT-B, the bottom, middle, and top segments contain 1, 10, and 1 transformer layers, respectively. The middle segment consists of 2 sparse-dense token modules, including 1 sparse token generation transformer, 2 sparse transformers, 1 dense token recovery transformer, and 1 dense transformer. As poolingformers and sparse transformers can interfere with the function of position embedding, we reintroduce sine-cosine position embedding at each stage of dense token generation. For DiT-XL, we use the checkpoint from the official DiT repository, while for DiT-B, we utilize the checkpoint provided by [40]. Following prior works, we sample 50,000 images to compute the Fréchet Inception Distance (FID) [17] using the ADM TensorFlow evaluation suite [12], along with the Inception Score (IS) [46], sFID [38], and Precision-Recall metrics [23]. Classifier-free guidance [19] (CFG) is set to 1.5 for evaluation and 4.0 for visualization. Throughput is evaluated with a batch size of 128 on an Nvidia A100 GPU.

Main results. The results of our approach on the class-conditional image generation task using ImageNet are presented in Table 1. We evaluate our method on two model sizes, DiT-B and DiT-XL, and at two resolutions, 256×256 and 512×512. At pruning rates $r \in [0.61, 0.86]$, SparseDiT-XL achieves a 43% reduction in FLOPs and an 87% improvement in inference speed, with only a 0.11 increase in FID score. This indicates significant redundancy within DiT, as using only about 25% tokens in specific layers maintains similar performance levels. By increasing the token count, we observe slight improvements over baseline performance with 145% improvement in throughput. The impact of our approach is even more pronounced at a resolution of 512×512. With a higher pruning rate than the 256×256 resolution, our method yields a superior performance-efficiency trade-off. By pruning over 90% of the tokens, we achieve a 55% reduction in FLOPs and a 175% increase in throughput, with only a 0.09 increase in FID score. When reducing the pruning rate further, our results slightly surpass the baseline, delivering a 145% throughput improvement. ToMeSD [5] is the first approach to reduce tokens. However, it applies token merging and recovery at every layer in a generic manner, resulting in a significant performance drop. For example, with a 0.1 merging rate on DiT-XL, ToMeSD achieves an FID score of 14.74, which is substantially worse than our result. A detailed comparison between other methods and our method is provided in Appendix A.1.

From 256×256 to 512×512, FLOPs increase by only 4.4 times, whereas the speed declines by a factor of 6.3, demonstrating that the primary speed bottleneck in DiT architecture is the number of tokens. Our method effectively reduces token count, resulting in significant gains, particularly for high-resolution content generation. For example, we achieve a 55% reduction in FLOPs at 512×512 while enhancing speed by 175%. Compared to other methods, our real-world speed improvements far exceed those achieved with similar reductions in FLOPs.

Table 3: Text-to-image generation on SAM dataset. $r \in [0.61, 0.86]$.

Model	FLOPs (G)	Throughput(img/s)	FID ↓
PixArt-α	148.73	0.414	4.53
Ours	91.62 (-38%)	0.701 (+69%)	4.29

Visualization results. To further validate the effectiveness of our method, we visualize some samples generated from SparseDiT-XL at a 256×256 resolution with a pruning rate $r \in [0.61, 0.86]$ (-43% FLOPs) and compare them with DiT-XL in Figure 4. Each cell in the same position corresponds to images generated from the same random seed and class. Since our model is fine-tuned from the pre-trained DiT, the overall styles and structures of the two subfigures are similar. The images generated by SparseDiT still retain rich high-frequency details. Furthermore, SparseDiT images demonstrate more precise structure, as evidenced by the accurate depiction of the "golden retriever's" nose and the "macaw's" eyes, which appear misplacement or missing in the images generated by DiT-XL. We provide samples at a resolution of 512×512 in Appendix.

4.2 Class-conditional video generation

Experimental settings. We conduct experiments at a resolution of 256×256 on four public datasets: FaceForensics [45], SkyTimelapse [60], UCF101 [54], and Taichi-HD [50]. Latte comprises two distinct types of transformer blocks: spatial transformers, which focus on capturing spatial information, and temporal transformers, which capture temporal information. To accommodate this separation of spatial and temporal feature extraction, we adjust our model's schedule within the sparse-dense token module. Specifically, we employ two transformers to prune spatial tokens and temporal tokens separately. The sparse tokens are then processed by two sparse transformers, and finally two transformers are used to recover the temporal and spatial tokens. Dense transformers are discarded in this setup. All other model configurations are the same as in DiT, and we follow the original training settings and hyperparameters used for Latte. To evaluate performance, we sample 2,048 videos, each consisting of 16 frames, and measure the Fréchet Video Distance (FVD) [57]. Throughput is measured using a batch size of 2 clips on an Nvidia A100 GPU.

Main results. Table 2 presents the main results of our approach. Leveraging the additional temporal dimension in video data allows for a higher pruning rate. Our method achieves a 56% reduction in FLOPs while maintaining a competitive FVD score compared to the baseline, demonstrating its effectiveness in video generation tasks.

4.3 Text-to-image generation

Experiment setting. We further assess the effectiveness of our method on text-to-image generation, which presents a greater challenge compared to class-conditional image generation. We adopt PixArt-α [8], a text-to-image generation model built based on DiT as the base model. Our model is initialized using the official PixArt-α checkpoint pre-trained on SAM dataset [22] containing 10M images. We further fine-tune it with our method on a subset of SAM dataset including 1M images. In PixArt-α, the transformer architecture includes two types of attention layers: a self-attention layer and a cross-attention layer, which integrates textual information. We apply our method to the self-attention layers to either reduce or recover tokens.

For evaluation, we randomly select text prompts from SAM dataset and adopt 100-step IDDPM solver [39] to sample 30,000 images, with the FID score serving as the evaluation metric. Throughput is evaluated with a batch size of 128 on an Nvidia A100 GPU.

Main results. As shown in Table 3, our SparseDiT achieves an FID score comparable to the original PixArT-α with significantly accelerating the generation, showing that our method is effective for text-to-image generation task.

Visualization results. We visualize some samples of our method and compare them with original PixArt-α in Figure 5. The classifier-free guidance is set to 4.5. The results demonstrate that our method effectively maintains both image quality and semantic fidelity.

4.4 Ablation study

All the following ablation experiments of SparseDiT-XL are conducted on ImageNet at a 256×256 resolution.

(a) Images generated by PixArt-α (b) Images generated by SparseDiT

Figure 5: Generating images from PixArt-α (a) and SparseDiT (b). The captions in the same positions are identical. They are provided in Appendix A.3.

Table 4: Performance evaluation on different numbers of SDTMs. "NAN" indicates loss nan.

No. of SDTMs	1	2	3	4
FID $\downarrow$	NAN	3.86	2.51	2.38
FLOPs	68.74	67.00	69.97	68.05

Table 5: Combination our method with efficient samplers. The numbers represent FID scores.

Model	250-DDPM	25-DDIM	5-RFlow
DiT-XL	2.27	2.89	43.40
Ours	2.38	3.31	43.69

The number of SDTMs. The alternation between sparse and dense tokens is a key factor contributing to the success of our method. Sparse tokens capture global structures, while dense tokens capture detailed information. In SparseDiT, we adopt four SDTMs by default. Table 4 shows the effect of using different numbers of SDTMs. To isolate the impact of SDTM count, we maintain the same FLOPs across models by adjusting the number of sparse and dense transformers while varying only the number of SDTMs. In configurations with only one SDTM, the model is modified into a U-shaped architecture. The results indicate that fewer SDTMs reduce interactions between global and local information, resulting in weaker performance. Furthermore, reducing SparseDiT to a U-Net structure compromises training stability, leading to a collapse. We do not experiment with more than four SDTMs, as increasing SDTM count beyond four offers diminishing returns in FLOP reduction.

Combination with efficient samplers. Our SparseDiT is a model compression method that can be seamlessly integrated with efficient samplers, such as DDIM [51] and Rectified Flow (RFlow) [28]. Note that the RFlow variant of DiT was trained by us via 2,000K iterations. As shown in Table 5, when our method is combined with the 25-step DDIM and 5-step Rectified Flow samplers, it achieves approximately $18.7\times$ and $93.4\times$ improvements in inference speed, respectively, compared to the standard 250-step DDPM. Notably, our method does not introduce a significant performance gap relative to the baseline, demonstrating that it can be effectively combined with efficient samplers.

The number of poolingformers. In Table 6, we examine the impact of varying the number of poolingformers in the model. We keep the total number of transformers in the bottom and top segments constant across configurations. When using three poolingformers, performance declines substantially due to the global average pooling operator's limited ability to adaptively capture information. For configurations with one or two poolingformers, the models achieve same results. The poolingformer is more efficient and consume fewer parameters than the standard transformer; therefore, we default to using 2 poolingformers. Reducing the poolingformer count to zero results in training instability, suggesting that maintaining full-size tokens in the initial layers is crucial. This finding aligns with the observations in Figure 1b: in DiT, the first two attention maps exhibit a uniform distribution, allowing us to replace them with poolingformers, but further replacement disrupts information flow and reduces performance.

The effectiveness of timestep-wise pruning rate. To evaluate the effectiveness of the timestep-wise pruning rate strategy, we conduct ablation experiments, as shown in Table 7. In these experiments, we maintain the same number of FLOPs while varying the number of tokens along timesteps, either using a constant token count or dynamically adjusting the token count. The results demonstrate that dynamically tuning the pruning rate significantly improves performance. Additionally, Table 7 highlights an observation: the performance of the dynamic pruning strategy is particularly relative

Table 6: Performance evaluation on different numbers of poolingformers. "NAN" indicates loss nan during training.

No. of Poolingformers	0	1	2	3
No. of trans in top seg.	4	3	2	1
FID $\downarrow$	NAN	2.38	2.38	2.56
FLOPs	69.56	68.80	68.05	67.29

Table 7: Comparison of FID scores with and without timestep-wise pruning rate strategy.

FLOPs (G)	No. of Tokens	Dynamic	FID $\downarrow$
~ 68	8×8		2.48
	$6 \times 6 \sim 10 \times 10$	$\checkmark$	2.38
~ 61	6×6		2.63
	$4 \times 4 \sim 8 \times 8$	$\checkmark$	2.50

with the token count of the later stages of denoising. For instance, the configuration with token counts ranging from 4×4 to 8×8 only shows a 0.02 FID score increase compared to the configuration with a constant 8×8 token count.

5   Conclusion

Limitations.   The primary limitation of our method arises from the manually pre-defined its structure, containing the number of layers in each module and the number of sparse tokens.

In this paper, we introduce SparseDiT, a novel approach to enhancing the efficiency of DiT by leveraging token sparsity. By incorporating SDTM and a timestep-wise pruning rate in a strategically layered manner, SparseDiT significantly reduces computational complexity while preserving high-generation quality. Experimental results show substantial FLOP reductions with minimal performance degradation, demonstrating SparseDiT's effectiveness across multiple DiT-based models and datasets. This work marks a step forward in developing scalable and efficient diffusion models for high-resolution content generation, expanding the practical applications of DiT-based architectures.

References

[1] F. Bao, C. Li, J. Zhu, and B. Zhang. Analytic-DPM: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. In *International Conference on Learning Representations*, 2022.

[2] F. Bao, S. Nie, K. Xue, Y. Cao, C. Li, H. Su, and J. 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.

[3] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[4] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations*, 2023.

[5] D. Bolya and J. Hoffman. Token merging for fast stable diffusion. *CVPR Workshop on Efficient Deep Learning for Computer Vision*, 2023.

[6] T. Brooks, B. Peebles, C. Holmes, W. DePue, Y. Guo, L. Jing, D. Schnurr, J. Taylor, T. Luhman, E. Luhman, et al. Video generation models as world simulators, 2024.

[7] S. Chang, P. Wang, M. Lin, F. Wang, D. J. Zhang, R. Jin, and M. Z. Shou. Making vision transformers efficient from a token sparsification view. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6195–6205, 2023.

[8] J. Chen, J. Yu, C. Ge, L. Yao, E. Xie, Y. Wu, Z. Wang, J. Kwok, P. Luo, H. Lu, and Z. Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis, 2023.

[9] T. Chen, Y. Cheng, Z. Gan, L. Yuan, L. Zhang, and Z. Wang. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34:19974–19988, 2021.

[10] X. Chen, N. Liu, Y. Zhu, F. Feng, and J. Tang. Edt: An efficient diffusion transformer framework inspired by human-like sketching. *arXiv preprint arXiv:2410.23788*, 2024.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

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

[13] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.

[14] G. Fang, X. Ma, and X. Wang. Structural pruning for diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

[15] M. Fayyaz, S. A. Koohpayegani, F. R. Jafari, S. Sengupta, H. R. V. Joze, E. Sommerlade, H. Pirsiavash, and J. Gall. Adaptive token sampling for efficient vision transformers. In *European Conference on Computer Vision*, pages 396–414. Springer, 2022.

[16] S. Gao, P. Zhou, M.-M. Cheng, and S. Yan. Mdtv2: Masked diffusion transformer is a strong image synthesizer. *arXiv preprint arXiv:2303.14389*, 2023.

[17] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[18] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[19] J. Ho and T. Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[20] Y. Huang, J. Huang, Y. Liu, M. Yan, J. Lv, J. Liu, W. Xiong, H. Zhang, S. Chen, and L. Cao. Diffusion model-based image editing: A survey. *arXiv preprint arXiv:2402.17525*, 2024.

[21] D. Kim, C.-H. Lai, W.-H. Liao, N. Murata, Y. Takida, T. Uesaka, Y. He, Y. Mitsufuji, and S. Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *The Twelfth International Conference on Learning Representations*, 2024.

[22] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[23] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.

[24] L. Li, H. Li, X. Zheng, J. Wu, X. Xiao, R. Wang, M. Zheng, X. Pan, F. Chao, and R. Ji. Autodiffusion: Training-free optimization of time steps and architectures for automated diffusion model acceleration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7105–7114, 2023.

[25] X. Li, Y. Liu, L. Lian, H. Yang, Z. Dong, D. Kang, S. Zhang, and K. Keutzer. Q-diffusion: Quantizing diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17535–17545, 2023.

[26] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[27] E. Liu, X. Ning, Z. Lin, H. Yang, and Y. Wang. OMS-DPM: Optimizing the model schedule for diffusion probabilistic models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 21915–21936. PMLR, 2023.

[28] X. Liu, C. Gong, and Q. Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[29] J. Lou, W. Luo, Y. Liu, B. Li, X. Ding, W. Hu, J. Cao, Y. Li, and C. Ma. Token caching for diffusion transformer acceleration. *arXiv preprint arXiv:2409.18523*, 2024.

[30] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. 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.

[31] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022.

[32] S. Luo, Y. Tan, L. Huang, J. Li, and H. Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2023.

[33] X. Ma, G. Fang, and X. Wang. Deepcache: Accelerating diffusion models for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15762–15772, 2024.

[34] X. Ma, Y. Wang, G. Jia, X. Chen, Z. Liu, Y.-F. Li, C. Chen, and Y. Qiao. Latte: Latent diffusion transformer for video generation. *arXiv preprint arXiv:2401.03048*, 2024.

[35] C. Meng, R. Rombach, R. Gao, D. Kingma, S. Ermon, J. Ho, and T. Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14297–14306, 2023.

[36] L. Meng, H. Li, B.-C. Chen, S. Lan, Z. Wu, Y.-G. Jiang, and S.-N. Lim. Adavit: Adaptive vision transformers for efficient image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12309–12318, 2022.

[37] T. Moon, M. Choi, E. Yun, J. Yoon, G. Lee, and J. Lee. Early exiting for accelerated inference in diffusion models. In *ICML 2023 Workshop on Structured Probabilistic Inference {\&} Generative Modeling*, 2023.

[38] C. Nash, J. Menick, S. Dieleman, and P. W. Battaglia. Generating images with sparse representations. *arXiv preprint arXiv:2103.03841*, 2021.

[39] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[40] Z. Pan, B. Zhuang, D.-A. Huang, W. Nie, Z. Yu, C. Xiao, J. Cai, and A. Anandkumar. T-stitch: Accelerating sampling in pre-trained diffusion models with trajectory stitching. *arXiv preprint arXiv:2402.14167*, 2024.

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

[42] Y. Pu, Z. Xia, J. Guo, D. Han, Q. Li, D. Li, Y. Yuan, J. Li, Y. Han, S. Song, et al. Efficient diffusion transformer with step-wise dynamic attention mediators. *arXiv preprint arXiv:2408.05710*, 2024.

[43] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[44] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[45] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. Faceforensics: A large-scale video dataset for forgery detection in human faces. *arXiv preprint arXiv:1803.09179*, 2018.

[46] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[47] T. Salimans and J. Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[48] Y. Shang, Z. Yuan, B. Xie, B. Wu, and Y. Yan. Post-training quantization on diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1972–1981, 2023.

[49] A. Shih, S. Belkhale, S. Ermon, D. Sadigh, and N. Anari. Parallel sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

[50] A. Siarohin, S. Lathuilière, S. Tulyakov, E. Ricci, and N. Sebe. First order motion model for image animation. *Advances in neural information processing systems*, 32, 2019.

[51] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[52] Y. Song and P. Dhariwal. Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*, 2023.

[53] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

[54] K. Soomro, A. R. Zamir, and M. Shah. A dataset of 101 human action classes from videos in the wild. *Center for Research in Computer Vision*, 2(11):1–7, 2012.

[55] S. Tang, Y. Wang, C. Ding, Y. Liang, Y. Li, and D. Xu. Deediff: Dynamic uncertainty-aware early exiting for accelerating diffusion model generation. *arXiv preprint arXiv:2309.17074*, 2023.

[56] Z. Tang, J. Tang, H. Luo, F. Wang, and T.-H. Chang. Accelerating parallel sampling of diffusion models. In *Forty-first International Conference on Machine Learning*, 2024.

[57] T. Unterthiner, S. Van Steenkiste, K. Kurach, R. Marinier, M. Michalski, and S. Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

[58] F.-Y. Wang, Z. Huang, A. W. Bergman, D. Shen, P. Gao, M. Lingelbach, K. Sun, W. Bian, G. Song, Y. Liu, et al. Phased consistency model. *arXiv preprint arXiv:2405.18407*, 2024.

[59] Y. Wang, R. Huang, S. Song, Z. Huang, and G. Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. *Advances in neural information processing systems*, 34:11960–11973, 2021.

[60] W. Xiong, W. Luo, L. Ma, W. Liu, and J. Luo. Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2364–2373, 2018.

[61] X. Yang, D. Zhou, J. Feng, and X. Wang. Diffusion probabilistic model made slim. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 22552–22562, 2023.

[62] Z. Yang, J. Teng, W. Zheng, M. Ding, S. Huang, J. Xu, Y. Yang, W. Hong, X. Zhang, G. Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

[63] T. Yin, M. Gharbi, R. Zhang, E. Shechtman, F. Durand, W. T. Freeman, and T. Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6613–6623, 2024.

[64] H. You, C. Barnes, Y. Zhou, Y. Kang, Z. Du, W. Zhou, L. Zhang, Y. Nitzan, X. Liu, Z. Lin, E. Shechtman, S. Amirghodsi, and Y. C. Lin. Layer-and timestep-adaptive differentiable token compression ratios for efficient diffusion transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[65] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10819–10829, 2022.

[66] Z. Yuan, H. Zhang, L. Pu, X. Ning, L. Zhang, T. Zhao, S. Yan, G. Dai, and Y. Wang. Ditfastattn: Attention compression for diffusion transformer models. *Advances in Neural Information Processing Systems*, 37:1196–1219, 2025.

[67] W. Zhao, Y. Han, J. Tang, Z. Li, Y. Song, K. Wang, Z. Wang, and Y. You. A stitch in time saves nine: Small vlm is a precise guidance for accelerating large vlms. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19814–19824, 2025.

[68] W. Zhao, Y. Han, J. Tang, K. Wang, Y. Song, G. Huang, F. Wang, and Y. You. Dynamic diffusion transformer. *arXiv preprint arXiv:2410.03456*, 2024.

[69] W. Zhao, J. Tang, Y. Han, Y. Song, K. Wang, G. Huang, F. Wang, and Y. You. Dynamic tuning towards parameter and inference efficiency for vit adaptation. *Advances in Neural Information Processing Systems*, 37:114765–114796, 2024.

[70] H. Zheng, W. Nie, A. Vahdat, K. Azizzadenesheli, and A. Anandkumar. Fast sampling of diffusion models via operator learning. In *International conference on machine learning*, pages 42390–42402. PMLR, 2023.

[71] Y. Zhu, X. Liu, and Q. Liu. Slimflow: Training smaller one-step diffusion models with rectified flow. In *European Conference on Computer Vision*, pages 342–359. Springer, 2025.

[72] Y. Zhu, W. Zhao, A. Li, Y. Tang, J. Zhou, and J. Lu. Flowie: Efficient image enhancement via rectified flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13–22, 2024.

[73] Z. Zong, K. Li, G. Song, Y. Wang, Y. Qiao, B. Leng, and Y. Liu. Self-slimmed vision transformer. In *European Conference on Computer Vision*, pages 432–448. Springer, 2022.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All the methods and experiments are served for claims in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: We discuss them in Conclusion.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All assumptions are provided.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All the key information is provided to reproduce the results. The codes will be released soon. All the experiment settings are the same as the original models.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code will be released soon.

   Guidelines:
   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All the experiment settings are the same as the original models.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: This paper does not include statistical significance.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All key details are provided in the Experiment section and the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: We have checked it.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss potential societal impacts in the text.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [Yes]

Justification: We discuss safeguards in the Appendix.

Guidelines:

- The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have cited the original paper that produced the code package and datasets and their licenses.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: his paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A   Appendix

A.1   Comparison with other methods

Comparison with ToMeSD.   Recent studies [37, 68] have evaluated ToMeSD [5] on DiT models and reported significant performance drops, which are much lower than those achieved by our method. Our approach is fundamentally different from ToMeSD. While ToMeSD reduces and recovers tokens generically at every layer, our method specifically analyzes the local-global relationships unique to DiT models. Based on these insights, we strategically reduce and recover tokens. We have evaluated ToMeSD on DiT and compared it with our method, as shown in Table 8.

Comparison with DyDiT.   DyDiT [68] represents the current state-of-the-art approach for enhancing the efficiency of Diffusion Transformers (DiTs). In contrast to our method, which emphasizes token reduction, DyDiT implements pruning across tokens, attention heads, and channels. A comparative analysis of our method against DyDiT, as detailed in Table 9 using ImageNet 512×512 images, illustrates a substantial advantage in computational efficiency for our approach. Specifically, our method achieves a 46% reduction in FLOPs, markedly surpassing DyDiT's 29% reduction, coupled with a 145% increase in inference speed, significantly outperforming the 31% improvement achieved by DyDiT. The only trade-off is a minor increase in the Fréchet Inception Distance (FID), where our method's FID is 0.08 higher than that of DyDiT. Nevertheless, this difference in FID is nearly imperceptible to human observers in practical applications.

Comparison with TokenCache.   We compare our method with TokenCache [29] in Table 9. We adopt the standard DiT baseline 2.27, while TokenCache applies a baseline 2.24. TokenCache achieves a 32% speed gain with a slight FID increase. Our approach attains an 87% speed gain with a smaller FID increase, demonstrating superior efficiency and performance.

Comparison with Ditfastattn.   We compare our method with Ditfastattn [66] in Table 9. We adopt the standard DiT baseline 3.04, while Ditfastattn applies a baseline 3.16. Ditfastattn shows a substantial FID increase for speed gain, while our method results in better speed and less FID.

A.2   Training form scratch

We train our SparseDiT from scratch, reaching 400K training iterations. We compare our result with original DiT paper in Table 4. The results (cfg=1.0) are in Table 10.

A.3   Captions in Figure 5

Column 1: The image depicts a beach scene with a large body of water, such as a lake or ocean, and a sandy shoreline. The beach is filled with people, including a group of people swimming in the water.

Column 2: The image depicts a beautiful outdoor dining area with a large number of tables and chairs arranged in a row, overlooking a picturesque lake. The tables are covered with white tablecloths, and there are several umbrellas providing shade for the guests. The scene is set in a lush green field, with a large building in the background, possibly a hotel or a restaurant. The tables are arranged in a way that allows for an unobstructed view of the lake, creating a serene and relaxing atmosphere for the diners. The image has a stylish and elegant feel, with the attention to detail in the table arrangement and the choice of location contributing to a memorable dining experience.

Column 3: The image depicts a busy city street at night, with a group of people standing outside a restaurant and a bar. The scene is set in a European city, and the atmosphere is lively and bustling.

Column 4: The image features a vintage-style train, parked on a track surrounded by trees and grass. The train appears to be an old-fashioned steam engine, which is a type of locomotive powered by steam. The train is positioned in a park-like setting, with a tree-lined path nearby. The scene is set in a sunny day, creating a pleasant atmosphere. The image has a nostalgic and historical feel, evoking a sense of the past and the charm of old-time trains.

Table 8: Comparison of our method with ToMeSD on DiT models.

Model	Method	FID	Speed-up
DiT-B	ToMeSD	29.24	+20%
	Ours	8.23	+68%
DiT-XL	ToMeSD	14.74	+66%
	Ours	2.38	+87%

Table 9: Comparison of our method with DyDiT on DiT models.

Model	Resolution	Method	FLOPs	FID	IS	Speed-up
DiT-XL	512×512	DyDiT	375 (-29%)	2.88(-0.16)	-	+31%
		Ditfastattn	-	4.52(+1.36)	180.34	+98%
		Ours	286 (-46%)	2.96(-0.08)	242.4	+145%
DiT-XL	256×256	TokenCache	72.25(-39%)	2.37 (+0.13)	262.00	+32%
		Ours	68.05(-43%)	2.38 (+0.11)	276.39	+87%

A.4 Additional visualization

We provide additional visualizations at a resolution of 512×512 on SparseDiT-XL from Figure 6 to Figure 17. The class labels, including "arctic wolf", "volcano", "cliff drop-off", "balloon", "sulphur-crested cockatoo", "lion", "otter", "coral reef", "macaw", "red panda", ""husky", and "panda", correspond to the same cases presented in DiT. Readers can compare our results with those of DiT. Our samples demonstrate comparable image quality and fidelity. The classifier-free guidance scale is set to 4.0, and all samples shown here are uncurated.

Table 10: Comparison of our method with ToMeSD on DiT models.

Method	FID	Flops	Interations
DiT-XL	19.47	118.64	400K
Ours (from scratch)	15.11	68.05	400K

Figure 6: Uncurated 512×512 SparseDiT-XL samples.
Classifier-free guidance scale = 4.0
Class label = "arctic wolf" (270)

Figure 7: Uncurated 512×512 SparseDiT-XL samples.
Classifier-free guidance scale = 4.0
Class label = "volcano" (980)

Figure 8: Uncurated 512×512 SparseDiT-XL samples.
Classifier-free guidance scale = 4.0
Class label = "husky" (250)

Figure 9: Uncurated 512×512 SparseDiT-XL samples.
Classifier-free guidance scale = 4.0
Class label = "sulphur-crested cockatoo" (89)

Figure 10: Uncurated 512×512 SparseDiT-XL samples.
Classifier-free guidance scale = 4.0
Class label = "cliff drop-off" (972)

Figure 11: Uncurated 512×512 SparseDiT-XL samples.
Classifier-free guidance scale = 4.0
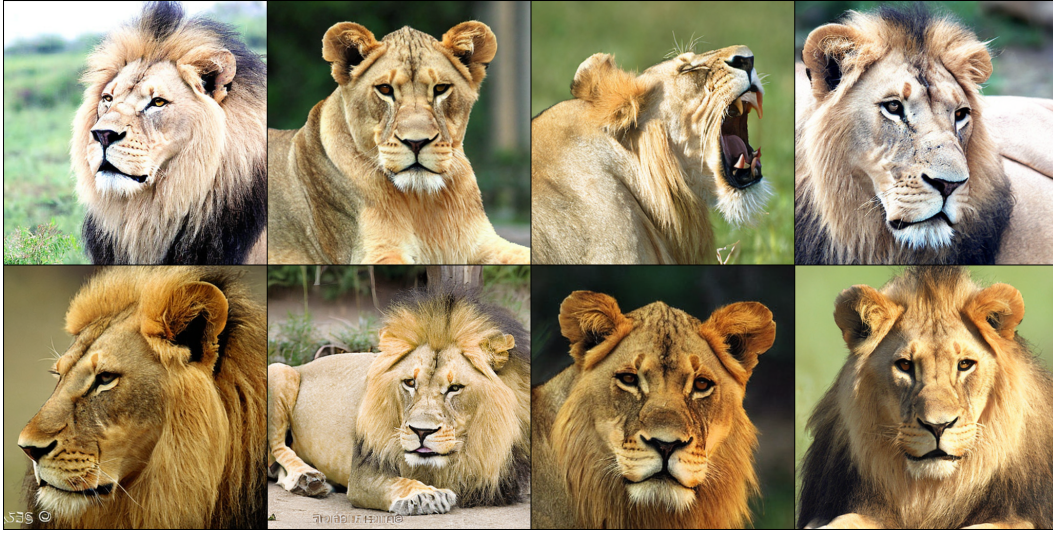Class label = "balloon" (417)

Figure 12: Uncurated 512×512 SparseDiT-XL samples.
Classifier-free guidance scale = 4.0
Class label = "lion" (291)

Figure 13: Uncurated 512×512 SparseDiT-XL samples.
Classifier-free guidance scale = 4.0
Class label = "otter" (360)

Figure 14: Uncurated 512×512 SparseDiT-XL samples.
Classifier-free guidance scale = 4.0
Class label = "red panda" (387)

Figure 15: Uncurated 512×512 SparseDiT-XL samples.
Classifier-free guidance scale = 4.0
Class label = "panda" (388)

Figure 16: Uncurated 512×512 SparseDiT-XL samples.
Classifier-free guidance scale = 4.0
Class label = "coral reef" (973)

Figure 17: Uncurated 512×512 SparseDiT-XL samples.
Classifier-free guidance scale = 4.0
Class label = "macaw" (88)