LazyDiT: Lazy Learning for the Acceleration of Diffusion Transformers

Xuan Shen[1*], Zhao Song[2], Yufa Zhou[3], Bo Chen[4], Yanyu Li[1], Yifan Gong[1], Kai Zhang[2], Hao Tan[2], Jason Kuen[2], Henghui Ding[5], Zhihao Shu[6], Wei Niu[6], Pu Zhao[1†], Yanzhi Wang[1†], Jiuxiang Gu[2†]

[1]Northeastern University
[2]Adobe Research
[3]University of Pennsylvania
[4]Middle Tennessee State University
[5]Fudan University
[6]University of Georgia
{shen.xu, p.zhao, yanz.wang}@northeastern.edu, wniu@uga.edu, jigu@adobe.com

Abstract

Diffusion Transformers have emerged as the preeminent models for a wide array of generative tasks, demonstrating superior performance and efficacy across various applications. The promising results come at the cost of slow inference, as each denoising step requires running the whole transformer model with a large amount of parameters. In this paper, we show that performing the full computation of the model at each diffusion step is unnecessary, as some computations can be skipped by lazily reusing the results of previous steps. Furthermore, we show that the lower bound of similarity between outputs at consecutive steps is notably high, and this similarity can be linearly approximated using the inputs. To verify our demonstrations, we propose the **LazyDiT**, a lazy learning framework that efficiently leverages cached results from earlier steps to skip redundant computations. Specifically, we incorporate lazy learning layers into the model, effectively trained to maximize laziness, enabling dynamic skipping of redundant computations. Experimental results show that LazyDiT outperforms the DDIM sampler across multiple diffusion transformer models at various resolutions. Furthermore, we implement our method on mobile devices, achieving better performance than DDIM with similar latency.

1 Introduction

Diffusion models (Ho, Jain, and Abbeel 2020; Rombach et al. 2022a; Song et al. 2020; Song and Ermon 2019; Dhariwal and Nichol 2021; Zhan et al. 2024b) have become dominant in image generation research, attributable to their remarkable performance. U-Net (Ronneberger, Fischer, and Brox 2015) is a widely used backbone in diffusion models, while transformers (Vaswani et al. 2017) are increasingly proving to be a strong alternative. Compared to U-Net, transformer-based diffusion models have demonstrated superior performance in high-fidelity image generation (Peebles and Xie 2023; Bao et al. 2023), and their efficacy extends to video generation as well (Lu et al. 2023; Chen et al. 2023; Lab and etc. 2024; Zheng et al. 2024). This highlights the versatility and potential of transformers in advancing generative tasks across different media. Despite the notable scalability advantages of transformers, diffusion transformers face major efficiency challenges. The high deployment costs and the slow inference speeds create the significant barriers to their practical applications (Zhan et al. 2024a,c, 2021; Wu et al. 2022; Li et al. 2022; Yang et al. 2023a), which motivates us to explore their acceleration methods.

The increased sampling cost in diffusion models stems from two main components: the numerous timesteps required and the computational expense associated with each inference step. To improve sampling efficiency, existing methods generally fall into two categories: reducing the total number of sampling steps (Song, Meng, and Ermon 2020; Liu et al. 2022; Bao et al. 2022; Zhan et al. 2024b) or lowering the computational cost per step (Yang et al. 2023b; He et al. 2023). Several works (Yin et al. 2024; Luo et al. 2024; Salimans and Ho 2022) employ distillation techniques to reduce the number of sampling steps. Conversely, works (Li et al. 2023b; Kim et al. 2023; Fang, Ma, and Wang 2023; Li et al. 2023a) utilize compression techniques to streamline diffusion models. Recently, some studies have introduced caching mechanisms into the denoising process (Ma, Fang, and Wang 2024; Wimbauer et al. 2023) to accelerate the sampling. However, previous compression approaches of diffusion models have primarily focused on optimizing U-Net, leaving transformer-based models largely unexplored.

Leveraging characteristic of uniquely structured, prior compression works (Zhang et al. 2024; Raposo et al. 2024; Fan, Grave, and Joulin 2019; Kong et al. 2022, 2023; Zhang et al. 2022; Li et al. 2023c; Zhao et al. 2024; Shen et al. 2024d,a,c,b, 2023b) have concentrated on techniques such as layer pruning and width pruning. However, we observe that removing certain layers results in a significant performance drop. This indicates the redundancy in diffusion transformers primarily occurs between sampling steps rather than the model architecture. This finding forms basis for exploring methods to reduce frequency of layer usage, aiming to decrease computational costs and accelerate the diffusion.

In this paper, we propose **LazyDiT**, a cache-based approach designed to dynamically reduce computational costs

[*]Work partially done during internship at Adobe Research
[†]Corresponding Author

Figure 1: Image generated by DiT-XL/2 in 512×512 and 256×256 resolutions when lazily skipping 50% computation. The upper rows display results from original model and the lower rows showcase outcomes of our method. Our method generates distinct lighting effects for background and color compared to the baseline, as demonstrated in dog and marmot, respectively.

and accelerate the diffusion process. We begin by analyzing the output similarity between the current and previous steps, identifying that the lower bound of this similarity is notably high during the diffusion process. Then, we delve deeper into the similarity using a Taylor expansion around the current input, revealing that the similarity can be linearly approximated. Building on the theoretical analysis, we implement a lazy learning framework by introducing linear layers before each Multi-Head Self-Attention (MHSA) and pointwise feedforward (Feedforward) module. These added layers are trained with the proposed lazy loss to learn whether the subsequent module can be lazily bypassed by leveraging the previous step's cache. Compared to the DDIM sampler, extensive experiments demonstrate that our method achieves superior performance with similar computational costs. As shown in Figure 1, by lazily skipping 50% of the computations, our method achieves nearly the same performance as the original diffusion process. We also profile the latency of the diffusion process on mobile devices to offer a detailed comparison with the DDIM sampler. Our results show our superior image generation quality than DDIM with similar latency. Our main contributions are summarized as follows,

- We explore the redundancy in diffusion process by evaluating the similarity between module outputs at consecutive steps, finding that the lower bound of the similarity is notably high.
- We establish that the lazy skip strategy can be effectively learned through a linear layer based on the Taylor expansion of similarity.
- We propose a lazy learning framework to optimize the diffusion process in transformer-based models by lazily bypassing computations using the previous step's cache.
- Experiments show that the proposed method achieves better performance than DDIM sampler. We further implement our method on mobile devices, showing that our method is a promising solution for real-time generation.

2 Related Work

Transformer-Based Diffusion Models. Recent works such as GenVit (Yang et al. 2022), U-Vit (Bao et al. 2023), DiT (Peebles and Xie 2023), LlamaGen (Sun et al. 2024), and MAR (Li et al. 2024a) have incorporated transformers (Vaswani et al. 2017) into diffusion models, offering a different approach compared to the traditional U-Net architecture. GenViT incorporates the ViT (Dosovitskiy et al. 2021; Li et al. 2024c,b) architecture into DDPM, while U-ViT further enhances this approach by introducing long skip connections between shallow and deep layers. DiT demonstrates the scalability of diffusion transformers, and its architecture has been further utilized for text-to-video generation tasks, as explored in works (OpenAI 2024). LlamaGen introduces autoregressive models to image generation, verifying the effectiveness of the 'next-token prediction' in this domain. Thus, it is crucial to explore efficient designs for those large models to accelerate the diffusion process.

Acceleration for Diffusion Models. High-quality image generation with diffusion models necessitates multiple sampling steps, leading to increased latency (Gong et al. 2024; Shen et al. 2023a). To enhance efficiency, DDIM (Song, Meng, and Ermon 2020) extends original DDPM to non-Markovian cases when DPM-Solver (Lu et al. 2022) advances the approximation of diffusion ODE solutions. Regarding the works that require fine-tuning, such as (Lin, Wang, and Yang 2024; Yin et al. 2024), they employ distillation techniques to effectively reduce the number of sampling steps. Additionally, reducing the computational work-

load for each diffusion step is a widely adopted, strategy to enhance the efficiency of the diffusion process. Various approaches have been explored, such as works (Fang, Ma, and Wang 2023; Castells et al. 2024; Wang et al. 2024; Zhang et al. 2024) that adopt weight pruning techniques, works (He et al. 2023; Li et al. 2023a) that employ quantization techniques, and even works (Kim et al. 2023; Li et al. 2023b) that redesign the architecture of diffusion models.

3 Methodology

3.1 Preliminaries

Notations. We use $\mathbb{E}[]$ to denote the expectation. We use $\mathbf{1}_n$ to denote a length-n vector where all the entries are ones. We use $x_{i,j}$ to denote the j-th coordinate of $x_i \in \mathbb{R}^n$. We use $\|x\|_p$ to denote the ℓ_p norm of a vector x. We use $\|A\|$ to denote the spectral norm for a matrix A. We use $a \circ b$ to denote the element-wise product of two vectors a, b. For a tensor $X \in \mathbb{R}^{B\times N\times D}$ and a matrix $U \in \mathbb{R}^{D\times d_1}$, we define $Y = X \cdot U \in \mathbb{R}^{B\times N\times d_1}$. For a matrix $V \in \mathbb{R}^{d_2\times B}$ and a tensor $X \in \mathbb{R}^{B\times N\times D}$, we define $Z = V \cdot X \in \mathbb{R}^{d_2\times N\times D}$. For a square matrix A, we use $\mathrm{tr}[A]$ to denote the trace of A. For two matrices X, Y, the standard inner product between matrices is defined by $\langle X, Y\rangle := \mathrm{tr}[X^\top Y]$. We use $U(a, b)$ to denote a uniform distribution. We use $\mathcal{N}(\mu, \sigma^2)$ to denote a Gaussian distribution. We define cosine similarity as $f(X, Y) = \frac{\mathrm{tr}[X^\top Y]}{\|X\|_F \cdot \|Y\|_F}$ for matrices X, Y.

Diffusion Formulation. Diffusion models (Ho, Jain, and Abbeel 2020; Song et al. 2020) operate by transforming a sample x from its initial state within a real data distribution $p_\beta(x)$ into a noisier version through diffusion steps. For a diffusion model $\epsilon_\theta(\cdot)$ with parameters θ, the training objective (Sohl-Dickstein et al. 2015) can be expressed as follows,

$$\min_\theta \mathop{\mathbb{E}}_{t\sim U[0,1], x\sim p_\beta(x), \epsilon\sim\mathcal{N}(0,I)} \|\epsilon_\theta(t, z_t) - \epsilon\|_2,$$

where t denotes the timestep; ϵ denotes the ground-truth noise; $z_t = \alpha_t \cdot x + \sigma_t \cdot \epsilon$ denotes the noisy data; α_t and σ_t are the strengths of signal and noise.

For comparison purposes, this paper adopts Denoising Diffusion Implicit Models (DDIM) (Song, Meng, and Ermon 2020) as sampler. The iterative denoising process from timestep t to the previous timestep t' is described as follows,

$$z_{t'} = \alpha_{t'} \cdot \frac{z_t - \sigma_t \epsilon_\theta(t, z_t)}{\alpha_t} + \sigma_{t'} \cdot \epsilon_\theta(t, z_t),$$

where $z_{t'}$ is iteratively fed to $\epsilon_\theta(\cdot)$ until t' becomes 0.

Latent Diffusion Models. The Latent Diffusion Model (LDM) (Rombach et al. 2022b) decreases computational demands and the number of steps with the latent space, which is obtained by encoding with a pre-trained variational autoencoder (VAE) (Sohl-Dickstein et al. 2015). Besides, the classifier-free guidance (CFG) (Ho and Salimans 2022) is adopted to improve quality as follows,

$$\widehat{\epsilon}_\theta(t, z_t, c) = w \cdot \epsilon_\theta(t, z_t, c) - (w - 1) \cdot \epsilon_\theta(t, z_t, c_\phi),$$

where $\epsilon_\theta(t, z_t, c_\phi)$ denotes unconditional prediction with null text; w denotes guidance scale which is used as control of conditional information and $w \geq 1$.

3.2 Similarity Establishment

Let B be the number of batches, N be the number of patches, D be the hidden dimension, T be the number of diffusion steps, and L be the number of model layers in diffusion transformers. Let $f(\cdot,\cdot) : \mathbb{R}^{B\times N\times D} \times \mathbb{R}^{B\times N\times D} \to [0, 1]$ be the function that estimate the similarity between two variables. Let $\mathcal{F}_l^\Phi(\cdot) : \mathbb{R}^{B\times N\times D} \to \mathbb{R}^{B\times N\times D}$ be the Multi-Head Self-Attention (MHSA) / pointwise feedforward (Feedforward) module at l-th layer where $\Phi \in \{\mathrm{attn}, \mathrm{feed}\}$ for $l \in [L]$. We use normalized $X_{l,t}^\Phi \in \mathbb{R}^{B\times N\times D}$ to denote the input hidden states with scaling factor a_t and shifting factor b_t at timestep t in the l-th layer for $t \in [T], l \in [L]$. We denote the output at l-th layer and t-th timestep as $Y_{l,t}^\Phi$.

Impact of Scaling and Shifting. As progressive denoising proceeds, the input difference between, $X_{l,t-1}^\Phi$ and $X_{l,t}^\Phi$, grows. In contrast, the application of scaling and shifting transformations introduces an alternate problem formulation, potentially affecting the input distance in a manner that requires a different analytical approach. In detail, the diffusion transformer architecture incorporates scaling and shifting mechanisms in the computation of both the MHSA and Feedforward modules, utilizing the embeddings of the timestep, $\mathrm{emd}(t) \in \mathbb{R}^D$, and the condition, $\mathrm{emd}(c) \in \mathbb{R}^D$. We define $y_t = \mathrm{SiLU}(\mathrm{emd}(t) + \mathrm{emd}(c)) \in \mathbb{R}^D$, with corresponding scaling and shifting factors defined as follows,

- Scaling factor: $a_t = W_{l,a} \cdot y_t + v_{l,a} \in \mathbb{R}^D$;
- Shifting factor: $b_t = W_{l,b} \cdot y_t + v_{l,b} \in \mathbb{R}^D$;

where $W_{l,a}, W_{l,b} \in \mathbb{R}^{D\times D}, v_{l,a}, v_{l,b} \in \mathbb{R}^D$ are the linear projection weight and bias, respectively.

Meanwhile, we define broadcasted matrices to represent the scaling and shifting factors, ensuring the alignment with the implementation of diffusion transformers as follows,

- Let $A_t \in \mathbb{R}^{N\times D}$ be defined as the matrix that all rows are a_t, i.e. $(A_t)_i := a_t$ for $i \in [N]$, and A_{t-1}, B_t, B_{t-1} can be defined as the same way.

Then, we deliver the demonstration showing that there exist y_t, y_{t-1} such that, after scaling and shifting, the distance between inputs $X_{l,t-1}^\Phi, X_{l,t}^\Phi$, defined in the Left Hand Side of the following Eq. (1), can be constrained within a small bound. Given a_t and b_t are both linear transformation of y_t, the problem reduces to demonstrating the existence of vectors a_t, b_t, a_{t-1}, and b_{t-1} that satisfy following conditions,

$$\|(A_{t-1} \circ X_{l,t-1}^\Phi + B_{t-1}) - (A_t \circ X_{l,t}^\Phi + B_t)\| \leq \eta, \quad (1)$$

where $\eta \in (0, 0.1)$. And Eq. (1) is equivalent as follows,

$$\|A \circ X_{l,t-1}^\Phi + B \circ X_{l,t}^\Phi + C\|_F \leq \eta, \quad (2)$$

where $A := A_{t-1}, B := -A_t, C := B_{t-1} - B_t$.

We identify that there exists a, b and c such that Eq. (2) holds, and the detailed demonstration and explanation are included in Lemma 12 at Appendix C.2. Subsequently, we generate the following theorem,

Theorem 1 (Scaling and shifting, informal version of Theorem 13 at Appendix C.2)**.** *There exist time-variant and condition-variant scalings and shiftings such that the distance between two inputs at consecutive steps for MHSA or Feedforward is bounded.*

Similarity Lower Bound. To leverage the cache from the previous step, we begin by investigating the cache mechanism in transformer-based diffusion models. This analysis focuses on the similarity between the current output and the preceding one, providing insights into the efficacy of reusing cached information. One typical transformer block consists of two primary modules: MHSA module and Feedforward module. Both modules are computationally expensive, making them significant contributors to the overall processing cost. Thus, we aim to examine the output similarities between the current and previous steps for both modules. By identifying cases of high similarity, we can skip redundant computations, thereby reducing the overall computational cost. In practice, we employ the cosine similarity $f(\cdot,\cdot)$ for the computation of similarity as follows,

$$f(Y^{\Phi}_{l,t-1}, Y^{\Phi}_{l,t}) = \frac{\mathrm{tr}[(Y^{\Phi}_{l,t-1})^{\top} \cdot Y^{\Phi}_{l,t}]}{\|Y^{\Phi}_{l,t-1}\|_F \cdot \|Y^{\Phi}_{l,t}\|_F}. \tag{3}$$

Inspired by the Lipschitz property (Trench 2013), we transform the similarity measure into a distance metric, defined as $\mathrm{Dist} := \|Y^{\Phi}_{l,t-1} - Y^{\Phi}_{l,t}\|$, to simplify the analysis of similarity variations. According to Fact 7 in Appendix B.2, similarity function $f(\cdot,\cdot)$ is further transformed as follows,

$$f(Y^{\Phi}_{l,t-1}, Y^{\Phi}_{l,t}) = 1 - \mathrm{Dist}/2.$$

For convenience, we further define the hidden states after scaling and shifting as $Z^{\Phi}_{l,t} := A_t \circ X^{\Phi}_{l,t} + B_t$. Meanwhile, building upon the Lemma H.5 of (Deng et al. 2023), we further derive the upper bounds for the distance Dist for either the MHSA or Feedforward modules as follows,

$$\mathrm{Dist} \leq C \cdot \|Z^{\Phi}_{l,t-1} - Z^{\Phi}_{l,t}\|. \tag{4}$$

where C is the Lipschitz constant related to the module.

Subsequently, with Theorem 1, we integrate Eq. (1) and derive the bound of the similarity as follows,

$$f(Y^{\Phi}_{l,t-1}, Y^{\Phi}_{l,t}) \geq 1 - \alpha,$$

for $\alpha := O(C^2\eta^2)$ and η is sufficiently small in practice.

Thus, we deliver Theorem 2 as below, which asserts that the lower bound of the output similarity between the two consecutive sampling steps is high.

Theorem 2 (Similarity lower bound, informal version of Theorem 18 at Appendix C.4)**.** *The lower bound of the similarity $f(Y^{\Phi}_{l,t-1}, Y^{\Phi}_{l,t})$ between the outputs at timestep $t-1$ and timestep t is high.*

Linear Layer Approximation. The similarity can be approximated using the inputs from either the current step $Z^{\Phi}_{l,t}$ or previous one $Z^{\Phi}_{l,t-1}$, due to its mathematical symmetry according to Eq. (3). We then apply the Taylor expansion around $Z^{\Phi}_{l,t}$ as follows,

$$\begin{aligned} & f(Y^{\Phi}_{l,t-1}, Y^{\Phi}_{l,t}) \\ = \ & \mathrm{tr}[(Y^{\Phi}_{l,t-1})^{\top} \cdot (0 + J \cdot Z^{\Phi}_{l,t} + O(1))], \end{aligned}$$

where J is the Jacobian matrix.

Through Taylor expansion, we identify that there exists a $W_l \in \mathbb{R}^{D \times D_{\mathrm{out}}}$ along with $Z^{\Phi}_{l,t}$ such that the similarity can be linearly approximated with certain error as follows,

$$\langle W^{\Phi}_l, Z^{\Phi}_{l,t} \rangle = f(Y^{\Phi}_{l,t-1}, Y^{\Phi}_{l,t}) + O(1),$$

where the detailed proof is included in Appendix C.5 Eq.(9).

Then, we generate the Theorem 3 as follows,

Theorem 3 (Linear layer approximation, informal version of Theorem 19 at Appendix C.5)**.** *The similarity function $f(\cdot,\cdot)$ can be approximated by a linear layer with respect to the current input, i.e. $f(Y^{\Phi}_{l,t-1}, Y^{\Phi}_{l,t}) = \langle W^{\Phi}_l, Z^{\Phi}_{l,t} \rangle$ where W^{Φ}_l is the weight of a linear layer for MHSA or Feedforward in the l-th layer of diffusion model.*

In our experiments, we utilize the lazy learning framework to obtain W^{Φ}_l, and we set $D_{\mathrm{out}} = 1$ to minimize computational cost. The details of this approach are explained in the following section.

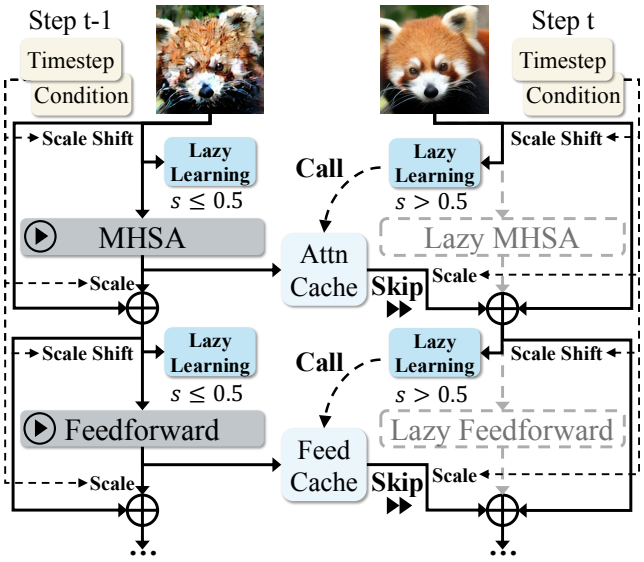

Figure 2: Overview framework. We skip the computation of MHSA or Feedforward by calling the previous step cache.

3.3 Lazy Learning

As illustrated in Figure 2, we incorporate lazy learning linear layers before each MHSA module and Feedforward module to learn the similarity. The MHSA module or Feedforward module is bypassed and replaced with the cached output from the previous step if the learned similarity is below 0.5. The input scale, input shift, output scale, and residual connections remain unchanged from the normal computation. The training details and the calculation of lazy ratio are outlined in the following paragraphs.

Training Forward. Assume we add linear layers with weights $W^{\Phi}_l \in \mathbb{R}^{D \times 1}$ for each module $\mathcal{F}^{\Phi}_l(\cdot)$ at l-th layer in the model. For input hidden states $X^{\Phi}_{l,t} \in \mathbb{R}^{B \times N \times D}$ for the module at l-th layer and t-th step, the similarity $s^{\Phi}_{l,t} \in \mathbb{R}^{B}$

of the module is computed as follows,

$$s_{l,t}^{\Phi} = \text{sigmoid}((Z_{l,t}^{\Phi} \cdot W_l^{\Phi}) \cdot \mathbf{1}_N).$$

We then define the forward pass of the MHSA module or Feedforward module at l-th layer and t-th step with the input $X_{l,t}^{\Phi}$ during the training progress as follows,

$$Y_{l,t}^{\Phi} = \text{diag}(\mathbf{1}_B - s_{l,t}^{\Phi}) \cdot \mathcal{F}_l^{\Phi}(Z_{l,t}^{\Phi}) + \text{diag}(s_{l,t}^{\Phi}) \cdot Y_{l,t-1}^{\Phi}.$$

Backward Loss. Alongside the diffusion loss for a given timestep t during training, we introduce a lazy loss to encourage the model to be more lazy—relying more on cached computations rather than diligently executing the MHSA modules or Feedforward modules, as follows,

$$\mathcal{L}_t^{\text{lazy}} = \rho^{\text{attn}} \cdot \frac{1}{B} \sum_{l=1}^{L} \sum_{b=1}^{B} (1 - (s_{l,t}^{\text{attn}})_b) + \rho^{\text{feed}} \cdot \frac{1}{B} \sum_{l=1}^{L} \sum_{b=1}^{B} (1 - (s_{l,t}^{\text{feed}})_b), \quad (5)$$

where the ρ^{attn} and ρ^{feed} denote the penalty ratio of MHSA module and Feedforward module, respectively.

We combine the lazy loss with diffusion loss and regulate ρ^{attn} and ρ^{feed} to control the laziness (i.e., number of skips with cache) of sampling with diffusion transformers.

Accelerate Sampling. After finishing the lazy learning with a few steps, we then accelerate the sampling during the diffusion process as follows,

$$Y_{l,t}^{\Phi} = \begin{cases} \mathcal{F}_l^{\Phi}(Z_{l,t}^{\Phi}), & s_{l,t}^{\Phi} \leq 0.5, \\ Y_{l,t-1}^{\Phi}, & s_{l,t}^{\Phi} > 0.5, \end{cases}$$

where $\Phi \in \{\text{attn}, \text{feed}\}$ can be either MHSA module or Feedforward module, and the skip occurs when $s_{l,t}^{\Phi} > 0.5$.

Then, the lazy ratio $\Gamma^{\Phi} \in \mathbb{Z}^B$ of MSHA or Feedforward for B batches during sampling is be computed as follows,

$$\Gamma^{\Phi} = \frac{1}{LT} \sum_{l=1}^{L} \sum_{t=1}^{T} \left\lceil s_{l,t}^{\Phi} - 0.5 \right\rceil.$$

4 Experimental Results

4.1 Experiment Setup

Model Family. We validate the effectiveness of our method on both the DiT (Peebles and Xie 2023) and LargeDiT (Zhang et al. 2023) model families. Specifically, our experiments utilize the officially provided models including DiT-XL/2 (256×256), DiT-XL/2 (512×512), Large-DiT-3B (256×256), and Large-DiT-7B (256×256).

Lazy Learning. We freeze the original model weights and introduce linear layers as lazy learning layers before each MHSA and Feedforward module at every diffusion step. For various sampling steps, these added layers are trained on the ImageNet dataset with 500 steps, with a learning rate of 1e-4 and using the AdamW optimizer. Following the training pipeline in DiT, we randomly drop some labels, assign a null token for classifier-free guidance, and set a global batch size of 256. The training is conducted on 8×NVIDIA A100 GPUs within 10 minutes.

Method	# of Step	Lazy Ratio	FID ↓	sFID ↓	IS ↑	Prec. ↑	Rec. ↑
DiT-XL/2 (256×256)							
DDIM	50	/	2.34	4.33	241.01	80.13	59.55
DDIM	40	/	2.39	4.28	236.26	80.10	59.41
Ours	50	20%	**2.37**	4.33	**239.99**	**80.19**	**59.63**
DDIM	30	/	2.66	4.40	234.74	79.85	58.96
Ours	50	40%	**2.63**	**4.35**	**235.69**	79.59	58.94
DDIM	25	/	2.95	4.50	230.95	79.49	58.44
Ours	50	50%	**2.70**	**4.47**	**237.03**	**79.77**	**58.65**
DDIM	20	/	3.53	4.91	222.87	78.43	57.12
Ours	40	50%	**2.95**	**4.78**	**234.10**	**79.61**	**57.99**
DDIM	14	/	5.74	6.65	200.40	74.81	55.51
Ours	20	30%	**4.44**	**5.57**	**212.13**	**77.11**	**56.76**
DDIM	10	/	12.05	11.26	160.73	66.90	51.52
Ours	20	50%	**6.75**	**8.53**	**192.39**	**74.35**	**52.43**
DDIM	7	/	34.14	27.51	91.67	47.59	46.83
Ours	10	30%	**17.05**	**13.37**	**136.81**	**62.07**	**50.37**
DiT-XL/2 (512×512)							
DDIM	50	/	3.33	5.31	205.01	80.59	55.89
DDIM	30	/	3.95	5.71	195.84	80.19	54.52
Ours	50	40%	**3.67**	**5.65**	**202.25**	79.80	**55.17**
DDIM	25	/	4.26	6.00	192.71	79.37	53.99
Ours	50	50%	**3.94**	**5.92**	**200.93**	**80.47**	**54.05**
DDIM	20	/	5.12	6.60	184.23	78.37	53.50
Ours	40	50%	**4.32**	**6.50**	**196.01**	**79.64**	53.19
DDIM	10	/	14.76	12.38	129.19	65.51	48.95
Ours	20	50%	**9.18**	**10.85**	**160.30**	**73.16**	**49.27**
DDIM	8	/	24.22	18.89	100.75	55.49	46.93
Ours	10	20%	**16.28**	**11.44**	**123.65**	**64.36**	**49.99**

Table 1: DiT model results on ImageNet (cfg=1.5). 'Lazy Ratio' indicates the percentage of skipped MHSA and Feedforward modules during diffusion process.

Penalty Regulation. We regulate the penalty ratios ρ^{attn} and ρ^{feed} for MHSA and Feedforward in Eq. (5) from 1e-7 to 1e-2. Both penalty ratios are kept identical in our experiments to optimize performance, as explained by the ablation study results shown in the lower of Figure 5.

Evaluation. To evaluate the effectiveness of our method, we primarily compare our method to the DDIM (Song, Meng, and Ermon 2020), varying the sampling steps from 10 to 50. Visualization results are generated with DiT-XL/2 model in 256×256 and 512×512 resolutions. For quantitative analysis, the 50,000 images are generated per trial with classifier-free guidance in our experiments. We adopt the Fréchet inception distance (FID) (Heusel et al. 2017), Inception Score (IS) (Salimans et al. 2016), sFID (Nash et al. 2021), and Precision/Recall (Kynkäänniemi et al. 2019) as the evaluation metrics. The computation cost as TMACs is calculated with the work (Zhu 2022).

Testing Bed. We implement our acceleration framework on mobile devices, specifically, we use OpenCL for mobile GPU backend. LazyDiT is built upon our existing DNN execution framework that supports extensive operator fusion

Method	# of Step	Lazy Ratio	FID ↓	sFID ↓	IS ↑	Prec. ↑	Rec. ↑
Large-DiT-3B (256×256)							
DDIM	50	/	2.10	4.36	263.83	80.36	59.55
DDIM	35	/	2.23	4.48	262.22	80.08	60.33
Ours	50	30%	**2.12**	**3.32**	**262.27**	**80.27**	60.01
DDIM	25	/	2.75	4.95	247.68	79.12	58.88
Ours	50	50%	**2.42**	**4.86**	**257.59**	**79.71**	**59.41**
DDIM	20	/	3.46	5.57	239.18	77.87	58.71
Ours	40	50%	**2.79**	**5.15**	**250.84**	**78.84**	**59.42**
DDIM	14	/	5.84	7.80	211.13	73.96	56.32
Ours	20	30%	**4.64**	**6.35**	**220.48**	**75.61**	**57.81**
DDIM	10	/	13.05	14.17	162.22	64.89	51.92
Ours	20	50%	**7.36**	**10.55**	**197.67**	**72.14**	**54.40**
DDIM	7	/	37.33	35.02	84.68	44.52	47.17
Ours	10	30%	**16.40**	**12.72**	**143.70**	**61.31**	**54.17**
Large-DiT-7B (256×256)							
DDIM	50	/	2.16	4.64	274.89	80.87	60.00
DDIM	35	/	2.29	4.83	267.31	80.42	59.21
Ours	50	30%	**2.13**	**4.49**	**267.37**	**80.55**	**60.76**
DDIM	25	/	2.76	5.36	259.07	79.33	58.76
Ours	50	50%	**2.53**	5.46	**265.26**	**80.48**	**58.88**
DDIM	20	/	3.32	6.05	247.94	78.51	57.78
Ours	40	50%	**2.90**	**6.01**	**257.47**	**79.67**	**57.97**
DDIM	14	/	5.66	8.80	218.50	74.57	55.18
Ours	20	30%	**4.97**	**7.30**	**220.99**	**75.04**	**57.60**
DDIM	10	/	12.70	15.93	166.66	65.27	52.67
Ours	20	50%	**7.00**	**11.42**	**206.57**	**72.61**	**55.14**
DDIM	7	/	36.57	39.76	84.54	44.69	47.44
Ours	10	30%	**16.83**	**22.76**	**143.14**	**61.05**	**50.23**

Table 2: Large-DiT model results on ImageNet (cfg=1.5). Full results included in Table 4 at Appendix A.1.

for various DNN structures. We also integrated other general DNN inference optimization methods similar to those in (Chen et al. 2018; Abadi et al. 2016), including memory layout and computation graph. Results are obtained using a smartphone with a Qualcomm Snapdragon 8 Gen 3, featuring a Qualcomm Kryo octa-core CPU, a Qualcomm Adreno GPU, and 16 GB of unified memory. Each result take 50 runs, with average results reported as variance is negligible.

4.2 Results on ImageNet

We present the results generated with DiT officially released models compared to DDIM in Table 1. Full results with more model sizes and lazy ratios are included in Table 5 of Appendix A.1. Due to the addition of lazy learning layers, the computational cost of our method is slightly higher than that of DDIM. Our experiments demonstrate that our method can perform better than the DDIM on DiT models with 256×256 and 512×512 resolutions. Particularly, for sampling steps fewer than 10, our method demonstrates a clear advantage over DDIM at both resolutions, highlighting the promise of our approach. For larger models with 3B and 7B parameters, we present the results in Table 2. Compared to the DiT-XL/2 model with 676M parameters, Large-DiT models with a few billion parameters exhibit more redundancy during the diffusion process. Full results for Large-DiT models are included in Table 4 at Appendix A.1. Experiments demonstrate that at 50% lazy ratio, our method significantly outperforms the approach of directly reducing sampling steps with DDIM. We further visualize the images generation results in Figure 1. We also compare with other cache-based method Learn2Cache (Ma et al. 2024) which adopts input independent cache strategy and requires full training on ImageNet, the results are in Table 7 at Appendix A.4. For each sampling step, Learn2Cache only has one cache strategy, whereas our method outperforms it with less training cost, demonstrating both the effectiveness and

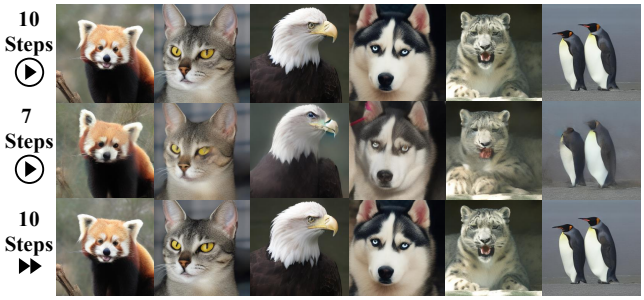

Figure 3: Image visualization generated by DiT-XL/2 model in 256×256 resolution on mobile. Images at the first and second rows are generated with 10 and 7 sampling steps. Images at the last row are generated with 30% lazy ratio.

Method	# of Step	Lazy Ratio	TMACs	IS ↑	Latency (s)
DiT-XL/2 (256×256)					
DDIM	50	/	5.72	241.01	21.62
DDIM	40	/	4.57	236.26	17.47
DDIM	25	/	2.86	230.95	11.33
Ours	50	50%	2.87	**237.03**	11.41
DDIM	20	/	2.29	222.87	9.29
DDIM	16	/	1.83	211.30	7.60
Ours	20	20%	1.83	**227.63**	7.67
DDIM	8	/	0.92	118.69	3.87
DDIM	7	/	0.80	91.67	3.54
Ours	10	30%	0.80	**136.81**	3.57
DiT-XL/2 (512×512)					
DDIM	50	/	22.85	205.01	75.09
DDIM	40	/	18.29	200.24	62.64
DDIM	25	/	11.43	192.71	41.37
Ours	50	50%	11.48	**200.93**	41.51
DDIM	13	/	5.94	156.82	23.71
DDIM	10	/	4.57	129.19	19.56
Ours	20	50%	4.59	**160.30**	19.77
DDIM	9	/	4.10	114.85	16.98
DDIM	8	/	3.66	100.75	15.69
Ours	10	20%	3.67	**123.65**	15.79

Table 3: Latency results on mobile devices with similar task performance or computation cost compared to the DDIM.

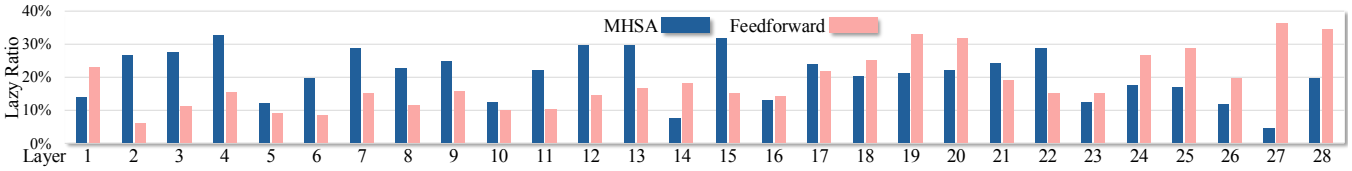

Figure 4: Visualization for the laziness in MHSA and Feedforward at each layer generated through DDIM 20 steps on DiT-XL.

the flexibility of our method.

4.3 Generation on Mobile

We present the latency profiling results on mobile devices in Table 3. Our method achieves better performance with less computation cost compared to DDIM. Additionally, when computational costs and latency are similar, our method significantly outperforms DDIM in terms of performance. Notably, with 10 sampling steps and a 30% lazy ratio, our method produces significantly higher image quality in 256×256 resolution than the DDIM sampler. Besides, we visualize the images generated on mobile in Figure 3. The images in the last row, generated with our method, exhibit higher quality compared to the second row, which are generated without the laziness technique under similar latency. Therefore, our method is especially beneficial for deploying diffusion transformers on mobile devices, offering a promising solution for real-time generation on edge platforms in the future. Meanwhile, the latency results tested on GPUs are included in Table 6 at Appendix A.2. Our method delivers much better performance with faster latency on GPUs, especially when the number of sampling steps are fewer than 10. Moreover, with almost the same latency, our method performs much better than DDIM.

4.4 Ablation Study

Individual Laziness. We perform ablation studies on the laziness of MHSA and Feedforward modules separately by regulating the corresponding penalty ratios to determine the maximum applicable laziness for each, thereby exploring the redundancy within both components. We present the results generated with DDIM 20 steps on DiT-XL/2 (256×256) in the upper figure in Figure 5. The analysis indicates that the maximum applicable lazy ratio is 30% for MHSA and 20% for Feedforward modules. The identification reveals that applying laziness individually to either MHSA or Feedforward network is not the most effective lazy strategy, which motivates us to apply the laziness to both modules simultaneously in our experiments.

Lazy Strategy. To optimize laziness in both MHSA and Feedforward modules for optimal performance, we fix the laziness in one module and regulate the penalty ratio of the other, varying the lazy ratio from 0% to 40%. Specifically, we separately fix 30% lazy ratio to MHSA or 20% lazy ratio to Feedforward modules, and analyzed the model performance by regulating the lazy ratio of another module with DDIM 20 steps on DiT-XL/2 (256×256). The results, as presented in the lower figure of Figure 5, reveal that the model achieves optimal performance when the same lazy ratio is

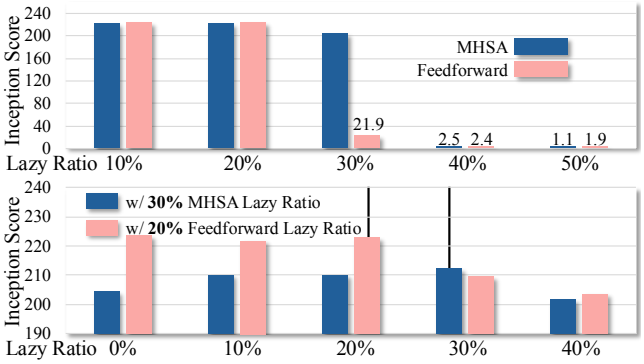

Figure 5: Upper figure: ablation for the generation performance with different individual laziness applied to each module independently. Lower figure: ablation for the generation performance with variant lazy ratio for one module and fixed lazy ratio for another module.

applied to both MHSA and Feedforward. Thus, we adopt the same penalty ratio for both modules in our experiments to achieve the best performance.

Layer-wise Laziness. To investigate the layer-wise importance during the diffusion process, we examined the laziness of each layer over 20 sampling steps with DiT-XL/2 model in 256×256 resolution with 8 images. The results, visualized in Figure 4, illustrate the layer-wise lazy ratio distribution and highlight key patterns in layer importance. The analysis reveals that, for MHSA, the latter layers are more critical, whereas for Feedforward layers, the initial layers hold greater importance. This is evidenced by the decreasing lazy ratio in MHSA and the increasing lazy ratio in MLP as going deeper. Moreover, all layers contribute to the process, as there is no such layer that has a 100% lazy ratio, meaning no layer is completely bypassed. Therefore, strategies such as removing layers or optimizing model structure are not applicable for transformer-based diffusion models.

5 Conclusion and Limitation

In this work, we introduce the LazyDiT framework to accelerate transformer-based diffusion models. We first show lower bound of similarity between consecutive steps is notably high. We then incorporate a lazy skip strategy inspired by the Taylor expansion of similarity. Experimental results validate the effectiveness of our method and we further implement our method on mobile devices, achieving better performance than DDIM. For the limitation, there is additional computation overhead for lazy learning layers.

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; et al. 2016. TensorFlow: A system for large-scale machine learning. In *OSDI 2016*, 265–283. USA: USENIX Association.

Bao, F.; Li, C.; Zhu, J.; and Zhang, B. 2022. Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. *arXiv preprint arXiv:2201.06503*.

Bao, F.; Nie, S.; Xue, K.; Cao, Y.; Li, C.; Su, H.; and Zhu, J. 2023. All are worth words: A vit backbone for diffusion models. In *CVPR*, 22669–22679.

Castells, T.; Song, H.-K.; Kim, B.-K.; and Choi, S. 2024. LD-Pruner: Efficient Pruning of Latent Diffusion Models using Task-Agnostic Insights. arXiv:2404.11936.

Chen, S.; Xu, M.; Ren, J.; et al. 2023. Gentron: Delving deep into diffusion transformers for image and video generation. *arXiv preprint arXiv:2312.04557*.

Chen, T.; Moreau, T.; Jiang, Z.; et al. 2018. TVM: An automated end-to-end optimizing compiler for deep learning. In *OSDI 2018*, 578–594.

Deng, Y.; Song, Z.; Xie, S.; and Yang, C. 2023. Unmasking transformers: A theoretical approach to data recovery via attention weights. *arXiv preprint arXiv:2310.12462*.

Dhariwal, P.; and Nichol, A. 2021. Diffusion models beat gans on image synthesis. *NeurIPS*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; et al. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv:2010.11929.

Fan, A.; Grave, E.; and Joulin, A. 2019. Reducing Transformer Depth on Demand with Structured Dropout. arXiv:1909.11556.

Fang, G.; Ma, X.; and Wang, X. 2023. Structural pruning for diffusion models. In *NeurIPS*.

Gong, Y.; Zhan, Z.; Jin, Q.; et al. 2024. E^2GAN: Efficient Training of Efficient GANs for Image-to-Image Translation. In *ICML*.

He, Y.; Liu, L.; Liu, J.; Wu, W.; Zhou, H.; and Zhuang, B. 2023. PTQD: Accurate Post-Training Quantization for Diffusion Models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *NeurIPS*.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *NeurIPS*.

Ho, J.; and Salimans, T. 2022. Classifier-Free Diffusion Guidance. arXiv:2207.12598.

Kim, B.-K.; Song, H.-K.; Castells, T.; and Choi, S. 2023. BK-SDM: Architecturally Compressed Stable Diffusion for Efficient Text-to-Image Generation. In *Workshop on Efficient Systems for Foundation Models @ ICML2023*.

Kong, Z.; Dong, P.; Ma, X.; et al. 2022. Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In *ECCV*.

Kong, Z.; Ma, H.; Yuan, G.; et al. 2023. Peeling the onion: Hierarchical reduction of data redundancy for efficient vision transformer training. In *AAAI*.

Kynkäänniemi, T.; Karras, T.; Laine, S.; Lehtinen, J.; and Aila, T. 2019. Improved Precision and Recall Metric for Assessing Generative Models. *CoRR*, abs/1904.06991.

Lab, P.-Y.; and etc., T. A. 2024. Open-Sora-Plan.

Li, T.; Tian, Y.; Li, H.; Deng, M.; and He, K. 2024a. Autoregressive Image Generation without Vector Quantization. *arXiv preprint arXiv:2406.11838*.

Li, X.; Liu, Y.; Lian, L.; Yang, H.; Dong, Z.; Kang, D.; Zhang, S.; and Keutzer, K. 2023a. Q-Diffusion: Quantizing Diffusion Models. In *ICCV*, 17535–17545.

Li, Y.; Wang, H.; Jin, Q.; Hu, J.; Chemerys, P.; Fu, Y.; Wang, Y.; Tulyakov, S.; and Ren, J. 2023b. SnapFusion: Text-to-Image Diffusion Model on Mobile Devices within Two Seconds. *arXiv preprint arXiv:2306.00980*.

Li, Y.; Yang, C.; Zhao, P.; et al. 2023c. Towards real-time segmentation on the edge. AAAI'23/IAAI'23/EAAI'23. AAAI Press. ISBN 978-1-57735-880-0.

Li, Y.; Zhang, Y.; Liu, S.; and Lin, X. 2024b. Pruning then Reweighting: Towards Data-Efficient Training of Diffusion Models. *arXiv preprint arXiv:2409.19128*.

Li, Y.; Zhao, P.; Ding, R.; Zhou, T.; Fei, Y.; Xu, X.; and Lin, X. 2024c. Neural architecture search for adversarial robustness via learnable pruning. *Frontiers in High Performance Computing*, 2: 1301384.

Li, Y.; Zhao, P.; Yuan, G.; et al. 2022. Pruning-as-Search: Efficient Neural Architecture Search via Channel Pruning and Structural Reparameterization. In *IJCAI*.

Lin, S.; Wang, A.; and Yang, X. 2024. SDXL-Lightning: Progressive Adversarial Diffusion Distillation. arXiv:2402.13929.

Liu, L.; Ren, Y.; Lin, Z.; and Zhao, Z. 2022. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*.

Lu, C.; Zhou, Y.; Bao, F.; Chen, J.; Li, C.; and Zhu, J. 2022. DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. arXiv:2206.00927.

Lu, H.; Yang, G.; Fei, N.; Huo, Y.; Lu, Z.; Luo, P.; and Ding, M. 2023. Vdt: General-purpose video diffusion transformers via mask modeling. *arXiv preprint arXiv:2305.13311*.

Luo, S.; Tan, Y.; Huang, L.; Li, J.; and Zhao, H. 2024. Latent Consistency Models: Synthesizing High-Resolution Images with Few-step Inference.

Ma, X.; Fang, G.; Mi, M. B.; and Wang, X. 2024. Learning-to-Cache: Accelerating Diffusion Transformer via Layer Caching. arXiv:2406.01733.

Ma, X.; Fang, G.; and Wang, X. 2024. DeepCache: Accelerating Diffusion Models for Free. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

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

OpenAI. 2024. Video generation models as world simulators. *https://openai.com/index/video-generation-models-as-world-simulators/*.

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

Raposo, D.; Ritter, S.; Richards, B.; et al. 2024. Mixture-of-Depths: Dynamically allocating compute in transformer-based language models. arXiv:2404.02258.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022a. High-resolution image synthesis with latent diffusion models. In *CVPR*, 10684–10695.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022b. High-Resolution Image Synthesis with Latent Diffusion Models. arXiv:2112.10752.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*.

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X. 2016. Improved techniques for training gans. *NeurIPS*.

Salimans, T.; and Ho, J. 2022. Progressive Distillation for Fast Sampling of Diffusion Models. In *International Conference on Learning Representations*.

Shen, X.; Dong, P.; Lu, L.; et al. 2024a. Agile-Quant: Activation-Guided Quantization for Faster Inference of LLMs on the Edge. In *AAAI*.

Shen, X.; Han, Z.; Lu, L.; et al. 2024b. HotaQ: Hardware Oriented Token Adaptive Quantization for Large Language Models. *TCAD*.

Shen, X.; Kong, Z.; Qin, M.; et al. 2023a. Data level lottery ticket hypothesis for vision transformers. *IJCAI*.

Shen, X.; Kong, Z.; Yang, C.; et al. 2024c. EdgeQAT: Entropy and Distribution Guided Quantization-Aware Training for the Acceleration of Lightweight LLMs on the Edge. *arXiv preprint arXiv:2402.10787*.

Shen, X.; Wang, Y.; Lin, M.; et al. 2023b. DeepMAD: Mathematical Architecture Design for Deep Convolutional Neural Network. In *CVPR*.

Shen, X.; Zhao, P.; Gong, Y.; et al. 2024d. Search for Efficient Large Language Models. In *NeurIPS*.

Sohl-Dickstein, J.; Weiss, E. A.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. arXiv:1503.03585.

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

Song, Y.; and Ermon, S. 2019. Generative modeling by estimating gradients of the data distribution. *NeurIPS*.

Song, Y.; Sohl-Dickstein, J.; Kingma, D. P.; Kumar, A.; Ermon, S.; and Poole, B. 2020. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Sun, P.; Jiang, Y.; Chen, S.; Zhang, S.; Peng, B.; Luo, P.; and Yuan, Z. 2024. Autoregressive Model Beats Diffusion: Llama for Scalable Image Generation. *arXiv preprint arXiv:2406.06525*.

Trench, W. F. 2013. Introduction to real analysis.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *NeurIPS*, 30.

Wang, K.; Chen, J.; Li, H.; Mi, Z.; and Zhu, J. 2024. SparseDM: Toward Sparse Efficient Diffusion Models. arXiv:2404.10445.

Wimbauer, F.; Wu, B.; Schoenfeld, E.; et al. 2023. Cache Me if You Can: Accelerating Diffusion Models through Block Caching. *arXiv preprint arXiv:2312.03209*.

Wu, Y.; Gong, Y.; Zhao, P.; et al. 2022. Compiler-aware neural architecture search for on-mobile real-time super-resolution. In *ECCV*, 92–111. Springer.

Yang, C.; Zhao, P.; Li, Y.; et al. 2023a. Pruning parameterization with bi-level optimization for efficient semantic segmentation on the edge. In *CVPR*, 15402–15412.

Yang, X.; Shih, S.-M.; Fu, Y.; Zhao, X.; and Ji, S. 2022. Your ViT is Secretly a Hybrid Discriminative-Generative Diffusion Model. arXiv:2208.07791.

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

Yin, T.; Gharbi, M.; Zhang, R.; Shechtman, E.; Durand, F.; Freeman, W. T.; and Park, T. 2024. One-step Diffusion with Distribution Matching Distillation. In *CVPR*.

Zhan, Z.; Gong, Y.; Zhao, P.; et al. 2021. Achieving on-mobile real-time super-resolution with neural architecture and pruning search. In *ICCV*, 4821–4831.

Zhan, Z.; Kong, Z.; Gong, Y.; et al. 2024a. Exploring Token Pruning in Vision State Space Models. In *NeurIPS*.

Zhan, Z.; Wu, Y.; Gong, Y.; et al. 2024b. Fast and Memory-Efficient Video Diffusion Using Streamlined Inference. In *NeurIPS*.

Zhan, Z.; Wu, Y.; Kong, Z.; et al. 2024c. Rethinking Token Reduction for State Space Models. In *EMNLP*, 1686–1697. Miami, Florida, USA: ACL.

Zhang, D.; Li, S.; Chen, C.; Xie, Q.; and Lu, H. 2024. LAPTOP-Diff: Layer Pruning and Normalized Distillation for Compressing Diffusion Models. arXiv:2404.11098.

Zhang, R.; et al. 2023. LLaMA-Adapter: Efficient Finetuning of Language Models with Zero-init Attention. *arXiv preprint arXiv:2303.16199*.

Zhang, Y.; Yao, Y.; Ram, P.; et al. 2022. Advancing model pruning via bi-level optimization. *NeurIPS*.

Zhao, P.; Sun, F.; Shen, X.; Yu, P.; Kong, Z.; Wang, Y.; and Lin, X. 2024. Pruning Foundation Models for High Accuracy without Retraining. In *Findings of EMNLP 2024*, 9681–9694. Miami, Florida, USA: ACL.

Zheng, Z.; Peng, X.; Yang, T.; Shen, C.; Li, S.; Liu, H.; Zhou, Y.; Li, T.; and You, Y. 2024. Open-Sora: Democratizing Efficient Video Production for All.

Zhu, L. 2022. pytorch-OpCounter. *https://github.com/Lyken17/pytorch-OpCounter*.