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

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. 2023c; Kim et al. 2023; Fang, Ma, and Wang 2023; Li et al. 2023b) 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.

In contrast to other architectures, transformers are uniquely structured with multiple layers that share a consistent design. Leveraging this characteristic, prior compression works (Zhang et al. 2024a; Raposo et al. 2024; Fan, Grave, and Joulin 2019; Kong et al. 2022, 2023; Zhang et al. 2022; Li et al. 2023d; 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 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 imple-

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.

ment a lazy learning framework by introducing linear layers before each Multi-Head Self-Attention (MHSA) and point-wise 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) 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 workload 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. 2024a; Zhang et al. 2024a) that adopt weight pruning techniques, works (He et al. 2023; Li et al. 2023b) that employ quantization techniques, and even works (Kim et al. 2023; Li et al. 2023c) 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 $\text{tr}[A]$ to denote the trace of A. For two matrices X, Y, the standard inner product between matrices is defined by $\langle X, Y \rangle := \text{tr}[X^\top Y]$. 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 $f(X, Y) = \frac{\text{tr}[X^\top Y]}{\|X\|_F \cdot \|Y\|_F}$ 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,

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

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 $\Phi \in \{\text{attn}, \text{feed}\}$ 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, $\text{emd}(t) \in \mathbb{R}^D$, and the condition, $\text{emd}(c) \in \mathbb{R}^D$. We define $y_t = \text{SiLU}(\text{emd}(t) + \text{emd}(c)) \in \mathbb{R}^D$, 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,

$$f(Y_{l,t-1}^{\Phi}, Y_{l,t}^{\Phi}) = \frac{\text{tr}[(Y_{l,t-1}^{\Phi})^{\top} \cdot Y_{l,t}^{\Phi}]}{\|Y_{l,t-1}^{\Phi}\|_F \cdot \|Y_{l,t}^{\Phi}\|_F}. \quad (3)$$

Inspired by the Lipschitz property (Trench 2013), we transform the similarity measure into a distance metric, defined as $\text{Dist} := \|Y_{l,t-1}^{\Phi} - Y_{l,t}^{\Phi}\|$, 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,

$$f(Y_{l,t-1}^{\Phi}, Y_{l,t}^{\Phi}) = 1 - \text{Dist}/2.$$

For convenience, we further define the hidden states after scaling and shifting as $Z_{l,t}^{\Phi} := A_t \circ X_{l,t}^{\Phi} + B_t$. 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,

$$\text{Dist} \leq C \cdot \|Z_{l,t-1}^{\Phi} - Z_{l,t}^{\Phi}\|. \quad (4)$$

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,

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

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 $f(Y_{l,t-1}^{\Phi}, Y_{l,t}^{\Phi})$ 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 $Z_{l,t}^{\Phi}$ or previous one $Z_{l,t-1}^{\Phi}$, due to its mathematical symmetry according to Eq. (3). We then apply the Taylor expansion around $Z_{l,t}^{\Phi}$ as follows,

$$f(Y_{l,t-1}^{\Phi}, Y_{l,t}^{\Phi})$$
$$= \text{tr}[(Y_{l,t-1}^{\Phi})^{\top} \cdot (0 + J \cdot Z_{l,t}^{\Phi} + O(1))],$$

where J is the Jacobian matrix.

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

$$\langle W_l^{\Phi}, Z_{l,t}^{\Phi} \rangle = f(Y_{l,t-1}^{\Phi}, Y_{l,t}^{\Phi}) + O(1),$$

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. $f(Y_{l,t-1}^{\Phi}, Y_{l,t}^{\Phi}) = \langle W_l^{\Phi}, Z_{l,t}^{\Phi} \rangle$ where W_l^{Φ} 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_l^{Φ}, and we set $D_{\text{out}} = 1$ to minimize computational cost. The details of this approach are explained in the following section.
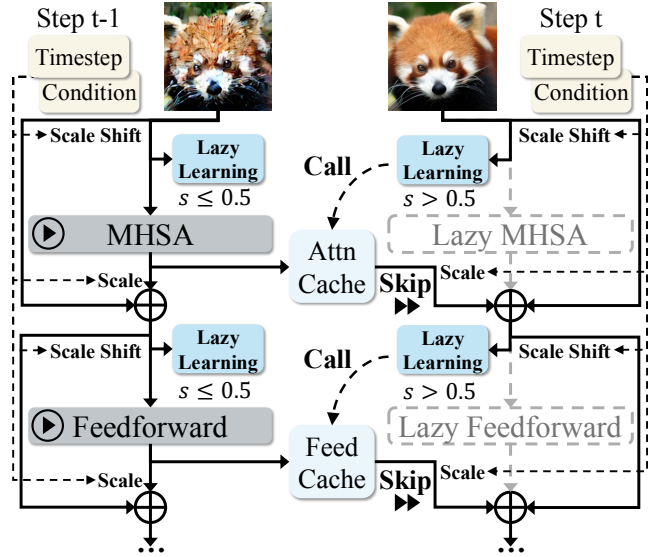

Figure 2: Overview framework. We skip the computation of MHSA or Feedforward by calling the previous step cache.

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 $W_l^{\Phi} \in \mathbb{R}^{D \times 1}$ for each module $\mathcal{F}_l^{\Phi}(\cdot)$ at l-th layer in the model. For input hidden states $X_{l,t}^{\Phi} \in \mathbb{R}^{B \times N \times D}$ for the module at l-th layer and t-th step, the similarity $s_{l,t}^{\Phi} \in \mathbb{R}^B$ 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,

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

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. 2024b) 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.

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.

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.

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 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, featur-

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.

ing 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

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.

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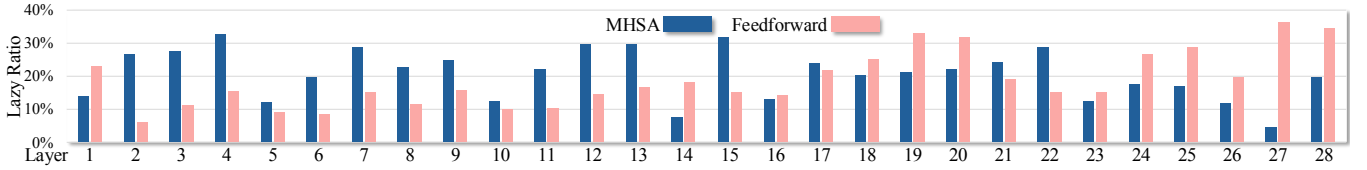
Figure 4: Visualization for the laziness in MHSA and Feedforward at each layer generated through DDIM 20 steps on DiT-XL.

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 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
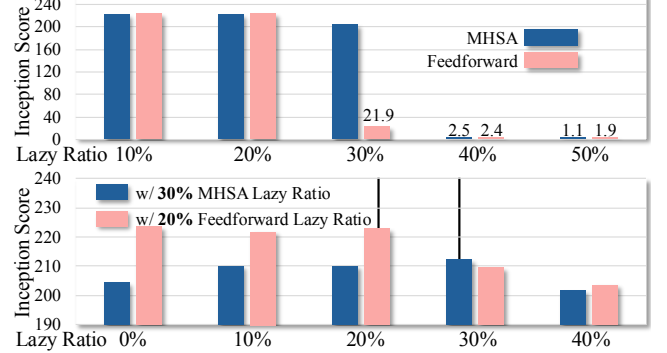

Figure 5: Upper figure: ablation for the generation performance with different individual laziness applied to each module independently. Lower figure: ablation for the generation performance with variant lazy ratio for one module and fixed lazy ratio for another module.

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, designed to accelerate the diffusion process in transformer-based diffusion models. Specifically, we first highlight redundancy in the diffusion process by showing that the lower bound of similarity between consecutive steps is notably high. Then, we design the lazy learning framework, which incorporates a lazy skip strategy inspired by the Taylor expansion of similarity. Experimental results validate the effectiveness of our method, indicating that it performs better than the DDIM sampler. We further implement our method on mobile devices, achieving better generation performance than DDIM sampler with lower latency. For the limitation, the paper also has some shortcomings, such as the additional computation overhead brought by the 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.

Alman, J.; and Song, Z. 2024. How to Capture Higher-order Correlations? Generalizing Matrix Softmax Attention to Kronecker Computation. In *The Twelfth International Conference on Learning Representations*.

Amini, A. A.; and Wainwright, M. J. 2009. High-dimensional analysis of semidefinite relaxations for sparse principal components. *The Annals of Statistics*, 37(5B).

Anstreicher, K. M. 2000. The Volumetric Barrier for Semidefinite Programming. *Math. Oper. Res.*, 25(3): 365–380.

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.

Bartlett, P. L.; Bousquet, O.; and Mendelson, S. 2005. Local rademacher complexities.

Belinkov, Y. 2022. Probing Classifiers: Promises, Shortcomings, and Advances. *Computational Linguistics*, 48(1): 207–219.

Bian, S.; Song, Z.; and Yin, J. 2023. Federated Empirical Risk Minimization via Second-Order Method. *arXiv preprint arXiv:2305.17482*.

Bottou, L.; and Bousquet, O. 2007. The tradeoffs of large scale learning. *Advances in neural information processing systems*, 20.

Brand, J. v. d. 2020. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 259–278. SIAM.

Brand, J. v. d.; Lee, Y. T.; Sidford, A.; and Song, Z. 2020. Solving tall dense linear programs in nearly linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, 775–788.

Brand, J. v. d.; Song, Z.; and Zhou, T. 2023. Algorithm and hardness for dynamic attention maintenance in large language models. *arXiv preprint arXiv:2304.02207*.

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.

Chang, C.-C.; and Lin, C.-J. 2001. Training v-support vector classifiers: theory and algorithms. *Neural computation*, 13(9): 2119–2147.

Chen, B.; Li, X.; Liang, Y.; Long, J.; Shi, Z.; and Song, Z. 2024a. Circuit Complexity Bounds for RoPE-based Transformer Architecture. *arXiv preprint arXiv:2411.07602*.

Chen, B.; Li, X.; Liang, Y.; Shi, Z.; and Song, Z. 2024b. Bypassing the Exponential Dependency: Looped Transformers Efficiently Learn In-context by Multi-step Gradient Descent. *arXiv preprint arXiv:2410.11268*.

Chen, B.; Liang, Y.; Sha, Z.; Shi, Z.; and Song, Z. 2024c. HSR-Enhanced Sparse Attention Acceleration. *arXiv preprint arXiv:2410.10165*.

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.

Chen, Y.; Zhong, R.; Zha, S.; Karypis, G.; and He, H. 2021. Meta-learning via language model in-context tuning. *arXiv preprint arXiv:2110.07814*.

Clark, K.; Khandelwal, U.; Levy, O.; and Manning, C. D. 2019. What Does BERT Look At? An Analysis of BERT's Attention. arXiv:1906.04341.

Cohen, M. B.; Lee, Y. T.; and Song, Z. 2021. Solving linear programs in the current matrix multiplication time. *Journal of the ACM (JACM)*, 68(1): 1–39.

d'Aspremont, A.; Ghaoui, L. E.; Jordan, M. I.; and Lanckriet, G. R. G. 2006. A direct formulation for sparse PCA using semidefinite programming. arXiv:cs/0406021.

Défossez, A.; and Bach, F. 2014. Constant step size least-mean-square: Bias-variance trade-offs and optimal sampling distributions. *arXiv preprint arXiv:1412.0156*.

Deng, Y.; Li, Z.; and Song, Z. 2023. Attention scheme inspired softmax regression. *arXiv preprint arXiv:2304.10411*.

Deng, Y.; Mahadevan, S.; and Song, Z. 2023. Randomized and deterministic attention sparsification algorithms for over-parameterized feature dimension. *arXiv preprint arXiv:2304.04397*.

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. *Advances in neural information processing systems*, 34: 8780–8794.

Diakonikolas, I.; Kamath, G.; Kane, D.; Li, J.; Moitra, A.; and Stewart, A. 2019. Robust Estimators in High Dimensions without the Computational Intractability. arXiv:1604.06443.

Dong, S.; Lee, Y. T.; and Ye, G. 2023. A Nearly-Linear Time Algorithm for Linear Programs with Small Treewidth: A Multiscale Representation of Robust Central Path. arXiv:2011.05365.

Dong, Y.; Hopkins, S. B.; and Li, J. 2019. Quantum Entropy Scoring for Fast Robust Mean Estimation and Improved Outlier Detection. arXiv:1906.11366.

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

Du, M.; He, F.; Zou, N.; Tao, D.; and Hu, X. 2023. Shortcut learning of large language models in natural language understanding. *Communications of the ACM*, 67(1): 110–120.

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 *Advances in Neural Information Processing Systems*.

Feldman, V.; Guruswami, V.; Raghavendra, P.; and Wu, Y. 2012. Agnostic learning of monomials by halfspaces is hard. *SIAM Journal on Computing*, 41(6): 1558–1590.

Frostig, R.; Ge, R.; Kakade, S. M.; and Sidford, A. 2015. Competing with the empirical risk minimizer in a single pass. In *Conference on learning theory*, 728–763. PMLR.

Gao, Y.; Mahadevan, S.; and Song, Z. 2023. An over-parameterized exponential regression. *arXiv preprint arXiv:2303.16504*.

Gao, Y.; Song, Z.; Wang, W.; and Yin, J. 2023a. A fast optimization view: Reformulating single layer attention in llm based on tensor and svm trick, and solving it in matrix multiplication time. *arXiv preprint arXiv:2309.07418*.

Gao, Y.; Song, Z.; Yang, X.; Zhang, R.; and Zhou, Y. 2023b. Fast quantum algorithm for attention computation. *arXiv preprint arXiv:2307.08045*.

Gao, Y.; Song, Z.; Yang, X.; and Zhou, Y. 2023c. Differentially private attention computation. *arXiv preprint arXiv:2305.04701*.

Gao, Y.; Song, Z.; and Yin, J. 2023. An iterative algorithm for rescaled hyperbolic functions regression. *arXiv preprint arXiv:2305.00660*.

Gao, Y.; Song, Z.; Zhang, R.; and Zhou, Y. 2024. Quantum Speedup for Spectral Approximation of Kronecker Products. *arXiv preprint arXiv:2402.07027*.

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

Gu, J.; Li, C.; Liang, Y.; Shi, Z.; and Song, Z. 2024a. Exploring the frontiers of softmax: Provable optimization, applications in diffusion model, and beyond. *arXiv preprint arXiv:2405.03251*.

Gu, J.; Li, C.; Liang, Y.; Shi, Z.; Song, Z.; and Zhou, T. 2024b. Fourier circuits in neural networks: Unlocking the potential of large language models in mathematical reasoning and modular arithmetic. *arXiv preprint arXiv:2402.09469*.

Gu, J.; Liang, Y.; Sha, Z.; Shi, Z.; and Song, Z. 2024c. Differential Privacy Mechanisms in Neural Tangent Kernel Regression. *arXiv preprint arXiv:2407.13621*.

Gu, Y.; and Song, Z. 2022. A Faster Small Treewidth SDP Solver. arXiv:2211.06033.

Gu, Y.; Song, Z.; and Zhang, L. 2023. A nearly-linear time algorithm for structured support vector machines. *arXiv preprint arXiv:2307.07735*.

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. *Advances in neural information processing systems*, 30.

Hewitt, J.; and Liang, P. 2019. Designing and Interpreting Probes with Control Tasks. arXiv:1909.03368.

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

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

Huang, B.; Jiang, S.; Song, Z.; Tao, R.; and Zhang, R. 2021. Solving SDP Faster: A Robust IPM Framework and Efficient Implementation. arXiv:2101.08208.

Jacq, A.; Ferret, J.; Pietquin, O.; and Geist, M. 2022. Lazy-MDPs: Towards Interpretable Reinforcement Learning by Learning When to Act. arXiv:2203.08542.

Jagielski, M.; Thakkar, O.; Tramèr, F.; Ippolito, D.; Lee, K.; Carlini, N.; Wallace, E.; Song, S.; Thakurta, A.; Papernot, N.; and Zhang, C. 2023. Measuring Forgetting of Memorized Training Examples. arXiv:2207.00099.

Jambulapati, A.; Li, J.; and Tian, K. 2020. Robust Sub-Gaussian Principal Component Analysis and Width-Independent Schatten Packing. arXiv:2006.06980.

Jiang, H.; Kathuria, T.; Lee, Y. T.; Padmanabhan, S.; and Song, Z. 2020a. A faster interior point method for semidefinite programming. In *2020 IEEE 61st annual symposium on foundations of computer science (FOCS)*, 910–918. IEEE.

Jiang, H.; Lee, Y. T.; Song, Z.; and wai Wong, S. C. 2020b. An Improved Cutting Plane Method for Convex Optimization, Convex-Concave Games and its Applications. arXiv:2004.04250.

Jiang, S.; Song, Z.; Weinstein, O.; and Zhang, H. 2020c. Faster dynamic matrix inverse for faster lps. *arXiv preprint arXiv:2004.07470*.

Jin, C.; Liu, L. T.; Ge, R.; and Jordan, M. I. 2018. On the local minima of the empirical risk. *Advances in neural information processing systems*, 31.

Joachims, T. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 217–226.

Johnson, R.; and Zhang, T. 2013. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26.

Kacham, P.; Mirrokni, V.; and Zhong, P. 2023. Polysketch-former: Fast transformers via sketches for polynomial kernels. *arXiv preprint arXiv:2310.01655*.

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.

Lee, Y. T.; Song, Z.; and Zhang, Q. 2019. Solving empirical risk minimization in the current matrix multiplication time. In *Conference on Learning Theory*, 2140–2157. PMLR.

Li, S.; Song, Z.; Xia, Y.; Yu, T.; and Zhou, T. 2023a. The closeness of in-context learning and weight shifting for softmax regression. *arXiv preprint arXiv:2304.13276*.

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.; Liang, Y.; Shi, Z.; Song, Z.; and Zhou, Y. 2024b. Fine-grained Attention I/O Complexity: Comprehensive Analysis for Backward Passes. *arXiv preprint arXiv:2410.09397*.

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

Li, X.; Long, J.; Song, Z.; and Zhou, T. 2024c. Fast Second-order Method for Neural Network under Small Treewidth Setting. In *2024 IEEE International Conference on Big Data (BigData)*. IEEE.

Li, Y.; Wang, H.; Jin, Q.; Hu, J.; Chemerys, P.; Fu, Y.; Wang, Y.; Tulyakov, S.; and Ren, J. 2023c. 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. 2023d. Towards real-time segmentation on the edge. AAAI'23/IAAI'23/EAAI'23. AAAI Press. ISBN 978-1-57735-880-0.

Li, Y.; Zhao, P.; Yuan, G.; et al. 2022. Pruning-as-Search: Efficient Neural Architecture Search via Channel Pruning and Structural Reparameterization. In *IJCAI-22*, 3236–3242.

Li, Z.; Song, Z.; Wang, Z.; and Yin, J. 2023e. Local Convergence of Approximate Newton Method for Two Layer Nonlinear Regression. *arXiv preprint arXiv:2311.15390*.

Li, Z.; Song, Z.; and Zhou, T. 2023. Solving regularized exp, cosh and sinh regression problems. *arXiv preprint arXiv:2303.15725*.

Liang, Y.; Long, J.; Shi, Z.; Song, Z.; and Zhou, Y. 2024a. Beyond Linear Approximations: A Novel Pruning Approach for Attention Matrix. *arXiv preprint arXiv:2410.11261*.

Liang, Y.; Sha, Z.; Shi, Z.; Song, Z.; and Zhou, Y. 2024b. Looped ReLU MLPs May Be All You Need as Practical Programmable Computers. *arXiv preprint arXiv:2410.09375*.

Liang, Y.; Sha, Z.; Shi, Z.; Song, Z.; and Zhou, Y. 2024c. Multi-Layer Transformers Gradient Can be Approximated in Almost Linear Time. *arXiv preprint arXiv:2408.13233*.

Liang, Y.; Shi, Z.; Song, Z.; and Zhou, Y. 2024d. Differential Privacy of Cross-Attention with Provable Guarantee. *arXiv preprint arXiv:2407.14717*.

Liang, Y.; Shi, Z.; Song, Z.; and Zhou, Y. 2024e. Tensor Attention Training: Provably Efficient Learning of Higher-order Transformers. *arXiv preprint arXiv:2405.16411*.

Liang, Y.; Shi, Z.; Song, Z.; and Zhou, Y. 2024f. Unraveling the Smoothness Properties of Diffusion Models: A Gaussian Mixture Perspective. *arXiv preprint arXiv:2405.16418*.

Lianke, Q.; Song, Z.; Zhang, L.; and Zhuo, D. 2023. An online and unified algorithm for projection matrix vector multiplication with application to empirical risk minimization. In *International Conference on Artificial Intelligence and Statistics*, 101–156. PMLR.

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*.

Moulines, E.; and Bach, F. 2011. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. *Advances in neural information processing systems*, 24.

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

Nemirovski, A.; Juditsky, A.; Lan, G.; and Shapiro, A. 2009. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4): 1574–1609.

Nesterov, Y. 1983. A method for solving the convex programming problem with convergence rate O (1/k2). In *Dokl akad nauk Sssr*, volume 269, 543.

Nesterov, Y. 2013. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media.

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.

Polyak, B. T.; and Juditsky, A. B. 1992. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4): 838–855.

Qin, L.; Song, Z.; and Sun, B. 2023. Is Solving Graph Neural Tangent Kernel Equivalent to Training Graph Neural Network? *arXiv preprint arXiv:2309.07452*.

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 *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, 234–241. Springer.

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

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

Shalev-Shwartz, S.; and Zhang, T. 2013. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(1).

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*.

Shrivastava, A.; Song, Z.; and Xu, Z. 2023. A Theoretical Analysis Of Nearest Neighbor Search On Approximate Near Neighbor Graph. *arXiv preprint arXiv:2303.06210*.

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. *Advances in neural information processing systems*, 32.

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*.

Song, Z.; Wang, W.; and Yin, J. 2023. A unified scheme of resnet and softmax. *arXiv preprint arXiv:2309.13482*.

Song, Z.; Xu, G.; and Yin, J. 2023. The Expressibility of Polynomial based Attention Scheme. *arXiv preprint arXiv:2310.20051*.

Song, Z.; Ye, M.; and Zhang, L. 2023. Streaming Semidefinite Programs: $O(\sqrt{n})$ Passes, Small Space and Fast Runtime. *arXiv preprint arXiv:2309.05135*.

Song, Z.; and Yu, Z. 2021. Oblivious sketching-based central path method for solving linear programming problems.

Sprechmann, P.; Jayakumar, S. M.; Rae, J. W.; Pritzel, A.; Badia, A. P.; Uria, B.; Vinyals, O.; Hassabis, D.; Pascanu, R.; and Blundell, C. 2018. Memory-based Parameter Adaptation. arXiv:1802.10542.

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*.

Tarzanagh, D. A.; Li, Y.; Thrampoulidis, C.; and Oymak, S. 2023. Transformers as support vector machines. *arXiv preprint arXiv:2308.16898*.

Tenney, I.; Das, D.; and Pavlick, E. 2019. BERT Rediscovers the Classical NLP Pipeline. arXiv:1905.05950.

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

van den Brand, J. 2020. Unifying Matrix Data Structures: Simplifying and Speeding up Iterative Algorithms. arXiv:2010.13888.

Vapnik, V. 1991. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4.

Vapnik, V. 2013. *The nature of statistical learning theory*. Springer science & business media.

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.

Vig, J.; and Belinkov, Y. 2019. Analyzing the Structure of Attention in a Transformer Language Model. arXiv:1906.04284.

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

Wang, Y.; Chen, Z.; Zhong, L.; Ding, Z.; Sha, Z.; and Tu, Z. 2023a. Dolfin: Diffusion Layout Transformers without Autoencoder. *arXiv preprint arXiv:2310.16305*.

Wang, Y.; Xu, H.; Zhang, X.; Chen, Z.; Sha, Z.; Wang, Z.; and Tu, Z. 2024b. OmniControlNet: Dual-stage Integration for Conditional Image Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7436–7448.

Wang, Z.; Sha, Z.; Ding, Z.; Wang, Y.; and Tu, Z. 2023b. To-kencompose: Grounding diffusion with token-level supervision. *arXiv preprint arXiv:2312.03626*.

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. 2024a. LAPTOP-Diff: Layer Pruning and Normalized Distillation for Compressing Diffusion Models. arXiv:2404.11098.

Zhang, R.; Han, J.; Liu, C.; Gao, P.; Zhou, A.; Hu, X.; Yan, S.; Lu, P.; Li, H.; and Qiao, Y. 2024b. Large-DiT. *https://github.com/Alpha-VLLM/LLaMA2-Accessory/tree/main/Large-DiT-ImageNet*.

Zhang, Y.; and Xiao, L. 2017. Stochastic primal-dual coordinate method for regularized empirical risk minimization. *Journal of Machine Learning Research*, 18(84): 1–42.

Zhang, Y.; Yao, Y.; Ram, P.; et al. 2022. Advancing model pruning via bi-level optimization. *NeurIPS*, 35: 18309–18326.

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*.

Appendix

A More Results

A.1 Full Results of Diffusion Transformers

We present the full results with Large-DiT models in Table 5 and DiT models including DiT-XL and DiT-L in Table 4. A diverse range of models with varying sampling steps has confirmed the effectiveness of our method. Besides, as the number of sampling steps decreases, our method proves to be even more effective compared to the standard 50-step sampling with the DDIM sampler. As there is no officially released DiT-L model, thus we pretrain it for one million steps. The results with DiT-L model here to show the generalization of our method to smaller models.

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	9	/	17.92	18.06	141.22	59.82	51.28
Ours	10	10%	15.30	12.55	150.35	62.68	53.86
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
DDIM	5	/	78.62	77.09	28.77	22.19	43.75
Ours	10	50%	45.99	48.67	67.63	38.97	47.13
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	16	/	4.45	7.28	232.91	76.30	56.42
Ours	20	20%	3.78	5.45	234.04	76.95	58.58
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	9	/	17.42	23.24	143.76	60.14	51.09
Ours	10	10%	16.47	20.50	145.04	61.33	49.84
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 4: LargeDiT model results on ImageNet (cfg=1.5).

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	35	/	2.47	4.29	235.18	79.63	59.20
Ours	50	30%	2.46	4.27	237.43	79.99	59.29
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	18	/	3.98	5.22	216.90	77.60	56.81
Ours	20	10%	3.39	4.81	229.32	79.21	57.66
DDIM	16	/	4.61	5.69	211.30	76.57	56.01
Ours	20	20%	3.45	4.83	227.63	79.16	58.00
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	9	/	16.52	14.34	141.14	62.39	50.53
Ours	10	10%	12.66	10.70	158.74	66.05	51.53
DDIM	8	/	23.17	19.04	118.69	56.25	48.52
Ours	10	20%	12.79	11.30	156.30	65.96	50.50
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
DDIM	6	/	52.02	41.77	60.63	36.21	44.95
Ours	10	40%	24.64	17.41	114.29	55.43	49.68
DDIM	5	/	63.87	77.61	32.24	23.70	43.31
Ours	10	50%	37.69	32.34	84.59	45.42	45.71
DiT-XL/2 (512×512)							
DDIM	50	/	3.33	5.31	205.01	80.59	55.89
DDIM	40	/	3.56	5.45	200.24	80.45	54.67
Ours	50	20%	3.54	5.41	201.87	80.41	55.27
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	14	/	7.87	8.34	162.48	74.14	52.21
Ours	20	30%	7.49	8.28	170.31	75.64	52.05
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
DiT-L/2 (256×256)							
DDIM	50	/	7.70	5.57	137.95	71.89	58.03
DDIM	40	/	8.14	5.83	135.74	71.16	57.81
DDIM	50	20%	7.90	5.51	138.69	71.63	57.17
DDIM	25	/	9.28	5.99	131.33	70.54	56.86
DDIM	50	50%	8.36	5.93	136.70	71.15	56.32
DDIM	20	/	10.24	6.58	128.11	69.44	56.17
DDIM	40	50%	9.36	6.77	132.20	70.29	55.23
DDIM	10	/	25.03	16.66	89.28	55.83	50.18
DDIM	20	50%	18.55	16.04	102.84	61.12	51.14
DDIM	8	/	39.57	28.08	66.18	45.07	47.20
DDIM	10	20%	27.82	16.71	83.67	53.03	51.24

Table 5: DiT model results on ImageNet (cfg=1.5).

Method	# of Step	Lazy Ratio	TMACs	IS ↑	Latency (s)
DiT-XL/2 (256×256)					
DDIM	50	/	5.72	241.01	7.39
DDIM	40	/	4.57	236.26	5.84
DDIM	25	/	2.86	230.95	3.65
Ours	50	50%	2.87	**237.03**	3.67
DDIM	20	/	2.29	222.87	2.88
DDIM	16	/	1.83	211.30	2.33
Ours	20	20%	1.83	**227.63**	2.33
DDIM	8	/	0.92	118.69	1.08
DDIM	7	/	0.80	91.67	0.98
Ours	10	30%	0.80	**136.81**	1.01
DiT-XL/2 (512×512)					
DDIM	50	/	22.85	205.01	38.42
DDIM	40	/	18.29	200.24	30.68
DDIM	25	/	11.43	192.71	19.15
Ours	50	50%	11.48	**200.93**	19.23
DDIM	13	/	5.94	156.82	9.74
DDIM	10	/	4.57	129.19	7.56
Ours	20	50%	4.59	**160.30**	7.60
DDIM	9	/	4.10	114.85	6.69
DDIM	8	/	3.66	100.75	6.08
Ours	10	20%	3.67	**123.65**	6.11

Table 6: Latency results on A5000 with 8 images per batch.

Method	# of Step	TMACs	FID ↓	IS ↑	Prec. ↑	Rec. ↑
DiT-XL/2 (256×256)						
DDIM	50	5.72	2.34	241.01	80.13	59.55
DDIM	40	4.57	2.39	236.26	80.10	59.41
Learn2Cache	50	4.36	2.39	238.89	80.18	58.59
Ours	50	4.58	**2.37**	**239.99**	**80.19**	**59.63**
DDIM	16	1.83	4.61	211.30	76.57	56.01
Learn2Cache	20	1.78	3.47	227.22	79.15	56.18
Ours	20	1.83	**3.45**	**227.63**	**79.16**	**58.00**
DDIM	9	1.03	16.52	141.14	62.39	50.53
Learn2Cache	10	1.04	12.77	156.39	66.01	52.26
Ours	10	1.03	**12.66**	**158.74**	**66.05**	51.53
DiT-XL/2 (512×512)						
DDIM	50	/	3.33	205.01	80.59	55.89
DDIM	30	13.71	3.95	195.84	80.19	54.52
Learn2Cache	50	14.19	3.76	201.13	80.08	54.21
Ours	50	13.77	**3.67**	**202.25**	79.80	**55.17**

Table 7: DiT model results on ImageNet (cfg=1.5) compared to Learn2Cache method (Ma et al. 2024).

A.5 More Visualization

More visualization is provided in Figure 7 in 512×512 resolution compared to DDIM sampler with 50% lazy ratio.

A.2 Latency on GPU

We present the latency profiling results with 8 images per batch in classifier-free guidance on a single A5000 GPU in Table 6. Our method achieves better performance with faster latency on GPUs. Meanwhile, for similar latency cost, our method outperform DDIM on the image generation quality.

A.3 Compare to Other Cache-Base Method

We further compare our method to the Learn2Cache method (Ma et al. 2024), we reproduce the training and evaluation with the given hyperparameter recipes. Notably, Learn2Cache requires a full epoch of training on ImageNet, which is significantly more intensive compared to our method, which only requires 500 training steps. Meanwhile, Learn2Cache does not support bigger laziness, offering only a single caching strategy for each specified number of sampling steps. The results are shown in Table 7. Our results achieve better performance than Learn2Cache method in most cases. Especially, our method performs much better with less computation cost when the resolution becomes 512×512.

A.4 Ablation for Skipping MHSA or Feedforward

We present the ablation study for skipping either MHSA or Feedforward layers individually, utilizing the sparse learning weights trained with the combined skipping of both components at same sparsity. The results are shown in Figure 6.

Figure 6: Visualization for the strategies of skipping MHSA or Feedforward only.

Figure 7: Visualization with 512×512 resolution with DDIM and our method with 50% lazy ratio.

B   Theoretical Preliminary

In this section, we provide the preliminary of our theoretical results.

B.1   Notations

For two vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$, we use $\langle x, y \rangle$ to denote the inner product between x, y, i.e., $\langle x, y \rangle = \sum_{i=1}^n x_i y_i$. We use e_i to denote a vector where only i-th coordinate is 1, and other entries are 0. For each $a, b \in \mathbb{R}^n$, we use $a \circ b \in \mathbb{R}^n$ to denote the Hardamard product, i.e. the i-th entry of $(a \circ b)$ is $a_i b_i$ for all $i \in [n]$. We denote the i-th row of a matrix X as X_i. 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 $x \in \mathbb{R}^n$, i.e. $\|x\|_1 := \sum_{i=1}^n |x_i|$, $\|x\|_2 := (\sum_{i=1}^n x_i^2)^{1/2}$, and $\|x\|_\infty := \max_{i \in [n]} |x_i|$. For $k > n$, for any matrix $A \in \mathbb{R}^{k \times n}$, we use $\|A\|$ to denote the spectral norm of A, i.e. $\|A\| := \sup_{x \in \mathbb{R}^n} \|Ax\|_2 / \|x\|_2$. We define $\|A\|_\infty := \max_{i \in [m], j \in [n]} |A_{i,j}|$ for $A \in \mathbb{R}^{m \times n}$. For a square matrix A, we use $\mathrm{tr}[A]$ to denote the trace of A, i.e., $\mathrm{tr}[A] = \sum_{i=1}^n A_{i,i}$. For two matrices X, Y, the standard inner product between matrices is defined by $\langle X, Y \rangle := \mathrm{tr}[X^\top Y]$. We define $\|X\|_F := (\sum_{i=1}^m \sum_{j=1}^n X_{i,j}^2)^{1/2}$ for a matrix $X \in \mathbb{R}^{m \times n}$.

B.2   Facts

In this section, we provide several facts we use.

Fact 4. *For a matrix $A \in \mathbb{R}^{m \times n}$, we have*

$$\|A\|_\infty \leq \|A\| \leq \sqrt{mn} \|A\|_\infty.$$

Fact 5. *For a matrix $A \in \mathbb{R}^{m \times n}$, we have*

$$\|A\| \leq \|A\|_F \leq \sqrt{k} \|A\|,$$

where k is the rank of A.

Fact 6 (Trace). *We have the following properties of trace:*

- $\|X\|_F^2 = \mathrm{tr}[X^\top X]$ *for matrix $X \in \mathbb{R}^{m \times n}$.*
- $\mathrm{tr}[X^\top Y] = \mathrm{tr}[XY^\top]$ *for matrices $X, Y \in \mathbb{R}^{m \times n}$.*
- $\mathrm{tr}[ABC] = \mathrm{tr}[CAB] = \mathrm{tr}[BCA]$ *for matrices $A, B, C \in \mathbb{R}^{m \times m}$.*

Fact 7. *For matrices $A, B \in \mathbb{R}^{m \times n}$, we have*

$$\|A - B\|_F^2 = \|A\|_F^2 + \|B\|_F^2 - 2\,\mathrm{tr}[A^\top B].$$

Proof. We can show

$$
\begin{aligned}
\|A - B\|_F^2 &= \mathrm{tr}[(A - B)^\top (A - B)] \\
&= \mathrm{tr}[A^\top A - A^\top B - B^\top A + B^\top B] \\
&= \mathrm{tr}[A^\top A] + \mathrm{tr}[B^\top B] - 2\,\mathrm{tr}[A^\top B] \\
&= \|A\|_F^2 + \|B\|_F^2 - 2\,\mathrm{tr}[A^\top B]
\end{aligned}
$$

where the first step follows from Fact 6, the second step follows from the basic algebra, the third and the last step follow from Fact 6. $\square$

Fact 8 (Taylor expansion for matrix). *Here, we provide the Taylor expansion for function $f(\cdot) : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$. For matrices $X \in \mathbb{R}^{m \times n}$, suppose we expand X around X_0. We have*

$$f(X) = f(X_0) + J \cdot (X - X_0) + O(\|X - X_0\|^2)$$

where J is a Jacobian matrix.

B.3   Definitions

In this section, we provide key definitions we use for the proofs. We define the attention module and feedforward module as follows.

Definition 9 (Self-attention module). *Given input matrix $X \in \mathbb{R}^{N \times D}$ and weight matrices $W_Q, W_K, W_V \in \mathbb{R}^{D \times D}$, (which represent the query, key and value weight matrices separately), we have self-attention module at l-th layer defined by:*

$$\mathcal{F}_l^{\mathrm{attn}}(X) := \underbrace{D^{-1}}_{N \times N} \underbrace{A}_{N \times N} \underbrace{V}_{N \times D}$$

where (1) $V := X W_V$, (2) $W := W_Q W_K^\top$, (3) $A := \exp(X W X^\top)$, and (4) $D := \mathrm{diag}(A \cdot \mathbf{1}_n)$.

Definition 10 (Feedforward module). *Given input matrix $X \in \mathbb{R}^{N \times D}$, we define the feedforward module at l-th layer as*

$$\mathcal{F}_l^{\mathrm{feed}}(X) := \underbrace{X}_{N \times D} \underbrace{W^{\mathrm{feed}}}_{D \times D}$$

where $W^{\mathrm{feed}} \in \mathbb{R}^{D \times D}$ is the linear weight for the feedforward layer.

C Theoretical Results

For simplicity of proofs, we assume the batch size $B = 1$. This will only affect the results by a constant factor of B. In this section, we provide the theoretical part of our paper.

C.1 More Related Work

Lazy Update. The concept of lazy update has emerged as a powerful strategy in addressing computational challenges. Lazy updating defers certain computations or updates until they are absolutely necessary, which can significantly reduce computational overhead in large-scale problems. This approach has applications to several fundamental theoretical frameworks such as Linear Programming (LP) and Semi-Definite Programming (SDP) (Anstreicher 2000; d'Aspremont et al. 2006; Amini and Wainwright 2009; Diakonikolas et al. 2019; Dong, Hopkins, and Li 2019; Cohen, Lee, and Song 2021; Jambulapati, Li, and Tian 2020; Jiang et al. 2020a; Gu and Song 2022; Song, Ye, and Zhang 2023). The application of lazy update strategies extends beyond LP and SDP to other important areas in computer science and machine learning. Dynamic maintenance of data structures and algorithms is one such area where lazy updates have proven valuable (Cohen, Lee, and Song 2021; Lee, Song, and Zhang 2019; Brand 2020; Jiang et al. 2020b; Brand et al. 2020; Jiang et al. 2020a,c; Song and Yu 2021; Dong, Lee, and Ye 2023; van den Brand 2020; Huang et al. 2021; Gu and Song 2022). This approach allows for efficient updates to solutions as input data changes, without the need for complete recomputation. In the realm of machine learning, Empirical Risk Minimization (ERM) is a fundamental principle that has benefited from lazy update techniques. ERM has been extensively studied in various contexts (Nesterov 1983; Vapnik 1991; Polyak and Juditsky 1992; Bartlett, Bousquet, and Mendelson 2005; Bottou and Bousquet 2007; Nemirovski et al. 2009; Moulines and Bach 2011; Feldman et al. 2012; Nesterov 2013; Johnson and Zhang 2013; Vapnik 2013; Shalev-Shwartz and Zhang 2013; Défossez and Bach 2014; Frostig et al. 2015; Zhang and Xiao 2017; Jin et al. 2018; Lee, Song, and Zhang 2019; Lianke et al. 2023; Bian, Song, and Yin 2023), with recent work focusing on improving its efficiency through lazy updates. Support Vector Machines (SVMs), a popular machine learning model, have also seen advancements through the application of lazy update strategies. Recent research has explored ways to optimize SVM training and inference using these techniques (Chang and Lin 2001; Joachims 2006; Tarzanagh et al. 2023; Brand, Song, and Zhou 2023; Li, Song, and Zhou 2023; Gao, Mahadevan, and Song 2023; Gu, Song, and Zhang 2023; Gao et al. 2023a), leading to improved performance in both time and space complexity.

Lazy Machine Learning. Lazy update strategies have been extensively explored in the context of machine learning to enhance algorithmic efficiency by reducing iterative loops. For instance, (Sprechmann et al. 2018) proposed a memory-based parameter adaptation (MbPA) method, leveraging lazy updates to optimize model parameters. In the domain of reinforcement learning, (Jacq et al. 2022) introduced lazy-MDPs, incorporating lazy update techniques to streamline the learning process. Furthermore, (Jagielski et al. 2023) investigated the interplay between forgetting and privacy protection during model training, highlighting the potential of lazy updates in balancing computational efficiency and data privacy. (Chen et al. 2021) proposed a meta-learning approach via in-context tuning, demonstrating the effectiveness of lazy update principles in adapting models to new tasks. Recently, (Du et al. 2023) provided a comprehensive overview of the shortcut learning problem in large language models for natural language understanding tasks, emphasizing the role of lazy update strategies in mitigating this challenge.

Attention Theory. Attention mechanisms have been extensively studied and developed over the years, becoming a fundamental component in various neural network architectures. (Deng, Li, and Song 2023; Li et al. 2023a; Gu et al. 2024a) investigate the softmax regression problem, while (Song, Ye, and Zhang 2023) propose and analyze the attention kernel regression problem. The rescaled hyperbolic functions regression is examined by (Gao, Song, and Yin 2023). Following this, (Song, Wang, and Yin 2023; Li et al. 2023e) delve into two-layer attention regression problems. (Gu et al. 2024b) demonstrate that attention layers in transformers learn two-dimensional cosine functions. Additionally, (Deng et al. 2023) explore data recovery using attention weights, and (Kacham, Mirrokni, and Zhong 2023; Song, Xu, and Yin 2023) investigate the replacement of the softmax unit with a polynomial unit. Moreover, some works theoretically explore variations or combinations of the attention mechanism with other techniques, such as quantum attention (Gao et al. 2023b, 2024), tensor attention (Alman and Song 2024; Liang et al. 2024e), and differentially private attention (Liang et al. 2024d; Gao et al. 2023c; Gu et al. 2024c) and other applications such as (Clark et al. 2019; Tenney, Das, and Pavlick 2019; Hewitt and Liang 2019; Vig and Belinkov 2019; Belinkov 2022; Wang et al. 2023b,a, 2024b; Brand, Song, and Zhou 2023; Liang et al. 2024f; Chen et al. 2024c,b,a; Shrivastava, Song, and Xu 2023; Qin, Song, and Sun 2023; Deng, Mahadevan, and Song 2023; Li et al. 2024c; Liang et al. 2024b; Li et al. 2024b; Liang et al. 2024c,e,a).

C.2 Effect of Scaling Factor

In this section, we examine the scaling factor used in DiT architecture. We first prove the vector case.

Lemma 11 (Scaling and shifting for one row). *If the following conditions hold:*

- *Let $x_1, x_2 \in \mathbb{R}^D$.*
- *Let $a, b, c \in \mathbb{R}^D$ and $a, b \neq 0$.*

Then, we can show that there exist a, b, c such that

$$\|a \circ x_1 + b \circ x_2 + c\|_2 \leq \eta$$

where $\eta \in (0, 0.1)$.

Proof. Let $a = \frac{0.5\eta}{\|x_1\|_2} \cdot \mathbf{1}_D$, $b = \frac{0.5\eta}{\|x_2\|_2} \cdot \mathbf{1}_D$, and $c = 0$.

Then, we have

$$
\begin{aligned}
&\|a \circ x_1 + b \circ x_2 + c\|_2 \\
&= \|\frac{0.5\eta \cdot x_1}{\|x_1\|_2} + \frac{0.5\eta \cdot x_2}{\|x_2\|_2} + 0\|_2 \\
&\leq \|\frac{0.5\eta \cdot x_1}{\|x_1\|_2}\|_2 + \|\frac{0.5\eta \cdot x_2}{\|x_2\|_2}\|_2 \\
&= \eta
\end{aligned}
$$

where the first step follows from the way we pick a, b, c, the second step follows from the triangle inequality, the last step follows from basic algebra. $\square$

It can be proven that for any matrix, there exist scaling and shifting factors that can adjust the input such that the distance between them is bounded.

Lemma 12 (Scaling and shifting). *If the following conditions hold:*

- *Let $X_1, X_2 \in \mathbb{R}^{N \times D}$.*
- *Let $a, b, c \in \mathbb{R}^D$ and $a, b \neq 0$.*
- *Let $A \in \mathbb{R}^{N \times D}$ be defined as the matrix that all rows are a, i.e. $A_i := a$ for $i \in [N]$, and B, C be defined as the same way.*
- *We define $M_1 := \max_{i \in [N]} \|X_{1,i}\|_2$ for X_1.*
- *We define $M_2 := \max_{i \in [N]} \|X_{2,i}\|_2$ for X_2.*

Then, we can show that there exist a, b, c such that

$$\|A \circ X_1 + B \circ X_2 + C\|_F \leq \eta$$

where $\eta \in (0, 0.1)$.

Proof. Let $a = \mathbf{1}_D \cdot \frac{0.5\eta}{NM_1}$, $b = \mathbf{1}_D \cdot \frac{0.5\eta}{NM_2}$, and $c = 0$. Let $\widetilde{A} = A \circ X_1$ and $\widetilde{B} = B \circ X_2$ and $\widetilde{C} = C$.

We define

$$\eta_0 := \eta/(3\sqrt{N}).$$

We can show

$$
\begin{aligned}
\|\widetilde{A}_i\|_2 &= \|a \circ X_{1,i}\|_2 \\
&= \frac{0.5\eta}{NM_1} \|X_{1,i}\|_2 \\
&\leq \frac{0.5\eta}{N} \\
&\leq \eta_0
\end{aligned}
\tag{6}
$$

where the first step follows from the definition of $\widetilde{A}$, the second step follows from the definition of a, the third step follows from the definition of M_1 and $X_{1,i}$, the last step follows from the definition of η_0.

We can show

$$
\begin{aligned}
\|\widetilde{B}_i\|_2 &= \|b \circ X_{2,i}\|_2 \\
&= \frac{0.5\eta}{NM_2} \|X_{2,i}\|_2 \\
&\leq \frac{0.5\eta}{N} \\
&\leq \eta_0
\end{aligned}
\tag{7}
$$

where the first step follows from the definition of $\widetilde{B}$, the second step follows from the definition of b, the third step follows from the definition of M_2 and $X_{2,i}$, the last step follows from the definition of η_0.

Because $c = 0$, we can show

$$\|\widetilde{C}_i\|_2 = 0.$$

Then, we can show

$$
\begin{aligned}
\|\widetilde{A} + \widetilde{B} + \widetilde{C}\|_F^2 &= \sum_{i=1}^{N} \|\widetilde{A}_i + \widetilde{B}_i + \widetilde{C}_i\|_2^2 \\
&\leq \sum_{i=1}^{N} 3(\|\widetilde{A}_i\|_2^2 + \|\widetilde{B}_i\|_2^2 + \|\widetilde{C}_i\|_2^2) \\
&\leq 3N \cdot 3\eta_0^2 \\
&\leq \eta^2
\end{aligned}
$$

where the first step follows from definition of Frobenius norm, the second step follows from Cauchy–Schwarz inequality, the third step follows from Eq.(6) and Eq.(7), and the last step follows from definition of η_0.

Taking the square root of the both side of the above equation, this can be further simplified as:

$$\|\widetilde{A} + \widetilde{B} + \widetilde{C}\|_F \leq \eta.$$

$\square$

Using the above idea, we can prove Theorem 13.

Theorem 13 (Effect of scaling and shifting, formal version of Theorem 1). *If the following conditions hold:*

- *Let $X_{l,t-1}^{\Phi}, X_{l,t}^{\Phi} \in \mathbb{R}^{N \times D}$.*
- *Let $a_t, a_{t-1}, b_t, b_{t-1} \in \mathbb{R}^D$.*
- *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} be defined as the same way.*

Then, we can show there exist $A_{t-1}, A_t, B_{t-1}, B_t$

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

where $\eta \in (0, 0.1)$.

Proof. We can show

$$
\begin{aligned}
&\|(A_{t-1} \circ X_{l,t-1}^{\Phi} + B_{t-1}) - (A_t \circ X_{l,t}^{\Phi} + B_t)\| \\
&= \|A_{t-1} \circ X_{l,t-1}^{\Phi} - A_t \circ X_{l,t}^{\Phi} + B_{t-1} - B_t\| \\
&\leq \|A_{t-1} \circ X_{l,t-1}^{\Phi} - A_t \circ X_{l,t}^{\Phi} + B_{t-1} - B_t\|_F \\
&:= \|A \circ X_{l,t-1}^{\Phi} + B \circ X_{l,t}^{\Phi} + C\|_F \\
&\leq \eta
\end{aligned}
$$

where the first step follows from basic algebra, the second step follows from Fact 5, the third step follows from we define $A := A_{t-1}, B := -A_t, C := B_{t-1} - B_t$, the last step follows from Lemma 12. $\square$

C.3 Lipschitz of Model Outputs

We first prove the Lipschitz for attention layer.

Lemma 14 (Lemma H.5 in (Deng et al. 2023)). *If the following conditions hold:*

- *Let $W, W_V \in \mathbb{R}^{D \times D}, X \in \mathbb{R}^{N \times D}$ be defined as in Definition 9.*
- *$R > 1$ be some fixed constant.*
- *Assume $\|W\| \leq R, \|W_V\| \leq R, \|X\| \leq R$.*

For $i \in [N], j \in [D]$, we have

$$|\mathcal{F}_l^{\mathrm{attn}}(X)_{i,j} - \mathcal{F}_l^{\mathrm{attn}}(\widetilde{X})_{i,j}| \leq 5R^4 \sqrt{ND} \cdot \|X - \widetilde{X}\|.$$

Using the previous results from (Deng et al. 2023), we can prove the Lipschitz for attention layer.

Lemma 15 (Lipschitz of attention). *If the following conditions hold:*

- *Let $W, W_V \in \mathbb{R}^{D \times D}, X \in \mathbb{R}^{N \times D}$ be defined as in Definition 9.*
- *$R > 1$ be some fixed constant.*
- *Assume $\|W\| \le R, \|W_V\| \le R, \|X\| \le R$.*

Then, we have the attention module (Definition 9) satisfying:

$$\|\mathcal{F}_l^{\mathrm{attn}}(X) - \mathcal{F}_l^{\mathrm{attn}}(\widetilde{X})\| \le 5R^4 N D \cdot \|X - \widetilde{X}\|.$$

Proof. Then we can show

$$\begin{aligned}
&\|\mathcal{F}_l^{\mathrm{attn}}(X) - \mathcal{F}_l^{\mathrm{attn}}(\widetilde{X})\| \\
&\le \sqrt{ND}\|\mathcal{F}_l^{\mathrm{attn}}(X) - \mathcal{F}_l^{\mathrm{attn}}(\widetilde{X})\|_\infty \\
&= \sqrt{ND} \max_{i \in [N], j \in [D]} |(\mathcal{F}_l^{\mathrm{attn}}(X) - \mathcal{F}_l^{\mathrm{attn}}(\widetilde{X}))_{i,j}| \\
&\le 5R^4 N D \|X - \widetilde{X}\|
\end{aligned}$$

where the first step follows from Fact 4, the second step follows the definition of $\| \cdot \|_\infty$, and the second step follows from Lemma 14. □

We then prove the Lipschitz for feedforward layer.

Lemma 16 (Lipschitz of feedforward). *If the following conditions hold:*

- *Let $X \in \mathbb{R}^{N \times D}$ and $W^{\mathrm{feed}} \in \mathbb{R}^{D \times D}$ be defined in Definition 10.*
- *$R > 1$ be some fixed constant.*
- *Assume $\|X\| \le R, \|W^{\mathrm{feed}}\| \le R$.*

  Then, we have the feedforward module (Definition 10) satisfying:

$$\|\mathcal{F}_l^{\mathrm{feed}}(X) - \mathcal{F}_l^{\mathrm{feed}}(\widetilde{X})\| \le R \cdot \| X - \widetilde{X}\|.$$

Proof. We can show

$$\begin{aligned}
\|\mathcal{F}_l^{\mathrm{feed}}(X) - \mathcal{F}_l^{\mathrm{feed}}(\widetilde{X})\| &= \|XW^{\mathrm{feed}} - \widetilde{X}W^{\mathrm{feed}}\| \\
&\le \|W^{\mathrm{feed}}\| \cdot \| X - \widetilde{X}\| \\
&\le R\| X - \widetilde{X}\|
\end{aligned}$$

where the first step follows from Definition 10, the second step follows from $\|AX\| \le \|A\| \cdot \|X\|$ for two matrices A, X, and the last step follows from $\|W^{\mathrm{feed}}\| \le R$. □

Combining the Lipschitz of two modules, we have the following lemma.

Lemma 17. *If the following conditions hold:*

- *Let $X \in \mathbb{R}^{N \times D}$.*
- *Let $W^{\mathrm{feed}} \in \mathbb{R}^{D \times D}$ be defined in Definition 10.*
- *Let $W, W_V \in \mathbb{R}^{D \times D}$ be defined as in Definition 9.*
- *$R > 1$ be some fixed constant.*
- *Assume $\|W^{\mathrm{feed}}\| \le R, \|W\| \le R, \|W_V\| \le R, \|X\| \le R$.*
- *$C = 5R^4 N D$ when $\Phi = \mathrm{attn}$, and $C = R$ when $\Phi = \mathrm{feed}$.*

Then, we have

$$\|\mathcal{F}_l^{\Phi}(X) - \mathcal{F}_l^{\Phi}(\widetilde{X})\| \le C \cdot \| X - \widetilde{X}\|.$$

Proof. Combining Lemma 15 and 16, we finish the proof. □

C.4   Similarity Lower Bound

In this section, we prove that the similarity of outputs between two diffusion timestep can be bounded.

Theorem 18 (Similarity lower bound, formal version of Theorem 2). *If the following conditions hold:*

- *Let $\alpha \in (0, 0.5)$.*
- *Let $X_{l,t}^{\Phi} \in \mathbb{R}^{N \times D}$, $W^{\mathrm{feed}} \in \mathbb{R}^{D \times D}$ be defined in Definition 10, and $W, W_V \in \mathbb{R}^{D \times D}$ be defined as in Definition 9.*
- *Assume $\|W^{\mathrm{feed}}\| \leq R, \|W\| \leq R, \|W_V\| \leq R, \|X\| \leq R$ for some fixed constant $R > 1$.*
- *Assume $R_1 \leq \|X_{l,t-1}^{\Phi} - X_{l,t}^{\Phi}\| \leq R_2$ for $R_1, R_2 \in \mathbb{R}$.*
- *$C = 5R^4 ND$ when $\Phi = \mathrm{attn}$, and $C = R$ when $\Phi = \mathrm{feed}$.*
- *We define $Y_{l,t}^{\Phi} := \mathcal{F}_l^{\Phi}(X_{l,t}^{\Phi})$ and assume $\|Y_{l,t}^{\Phi}\|_F = 1$ for any t.*

Then, the similarity between $Y_{l,t-1}^{\Phi}$ and $Y_{l,t}^{\Phi}$ can be bounded by:

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

Proof. Suppose we use cosine similarity. We derive a new form of f first.

$$
\begin{aligned}
& f(Y_{l,t-1}^{\Phi}, Y_{l,t}^{\Phi}) \\
&= \frac{\mathrm{tr}[Y_{l,t-1}^{\Phi} \cdot (Y_{l,t}^{\Phi})^{\top}]}{\|Y_{l,t-1}^{\Phi}\|_F \cdot \|Y_{l,t}^{\Phi}\|_F} \\
&= \frac{0.5(\|Y_{l,t-1}^{\Phi}\|_F^2 + \|Y_{l,t}^{\Phi}\|_F^2 - \|Y_{l,t-1}^{\Phi} - Y_{l,t}^{\Phi}\|_F^2)}{\|Y_{l,t-1}^{\Phi}\|_F \cdot \|Y_{l,t}^{\Phi}\|_F} \\
&= 1 - 0.5 \cdot \|Y_{l,t-1}^{\Phi} - Y_{l,t}^{\Phi}\|_F^2 \qquad (8)
\end{aligned}
$$

where the first step follows from the definition of cosine similarity, the second step follows from Fact 7, the third step follows from $\|Y_{l,t}^{\Phi}\|_F = 1$ for any t.

Then, we can show that

$$
\begin{aligned}
& f(Y_{l,t-1}^{\Phi}, Y_{l,t}^{\Phi}) \\
&= 1 - 0.5 \cdot \|Y_{l,t-1}^{\Phi} - Y_{l,t}^{\Phi}\|_F^2 \\
&\geq 1 - 0.5 \min\{N, D\} \cdot \|Y_{l,t-1}^{\Phi} - Y_{l,t}^{\Phi}\|^2 \\
&\geq 1 - 0.5 C^2 \min\{N, D\} \cdot \|X_{l,t-1}^{\Phi} - X_{l,t}^{\Phi}\|^2 \\
&\geq 1 - 0.5 C^2 R_2^2 \min\{N, D\} \\
&:= 1 - \alpha
\end{aligned}
$$

where the first step follows from Eq. (8), the second step follows from Fact 5, the third step follows from Lemma 17, the fourth step follows from $\|X_{l,t-1}^{\Phi} - X_{l,t}^{\Phi}\| \leq R_2$, and the last step follows from we define $\alpha := 0.5 C^2 R_2^2 \min\{N, D\}$. $\square$

C.5   Approximating Similarity Function

In this section, we prove that the similarity function can be approximated by a linear linear with certain error.

Theorem 19 (Linear layer approximation, formal version of Theorem 3). *If the following conditions hold:*

- *The similarity function f is assumed to be sufficiently smooth such that its derivatives up to at least the second order are well-defined and continuous over the range of interest.*
- *We define $Y_{l,t}^{\Phi} := \mathcal{F}_l^{\Phi}(X_{l,t}^{\Phi})$ and assume $\|Y_{l,t}^{\Phi}\|_F = 1$ for any t.*

There exist weights W_l of a linear layer in the l-th layer of diffusion model such that $f(Y_{l,t-1}^{\Phi}, Y_{l,t}^{\Phi}) = \langle W_l^{\Phi}, X_{l,t}^{\Phi} \rangle$.

Proof. Suppose we use cosine similarity. We have

$$
\begin{aligned}
& f(Y_{l,t-1}^{\Phi}, Y_{l,t}^{\Phi}) \\
& = \frac{\mathrm{tr}[Y_{l,t-1}^{\Phi} \cdot (Y_{l,t}^{\Phi})^{\top}]}{\|Y_{l,t-1}^{\Phi}\|_F \cdot \|Y_{l,t}^{\Phi}\|_F} \\
& = \mathrm{tr}[Y_{l,t-1}^{\Phi} \cdot (Y_{l,t}^{\Phi})^{\top}] \\
& = \mathrm{tr}[(Y_{l,t-1}^{\Phi})^{\top} \cdot Y_{l,t}^{\Phi}] \\
& = \mathrm{tr}[(Y_{l,t-1}^{\Phi})^{\top} \cdot (0 + J \cdot (X_{l,t}^{\Phi} - 0) + O(1))] \\
& = \mathrm{tr}[(Y_{l,t-1}^{\Phi})^{\top} \cdot J \cdot X_{l,t}^{\Phi}] + O(1) \\
& = \langle (Y_{l,t-1}^{\Phi})^{\top} \cdot J, X_{l,t}^{\Phi} \rangle + O(1) \\
& := \langle W_l^{\Phi}, X_{l,t}^{\Phi} \rangle + O(1)
\end{aligned}
\tag{9}
$$

where the first step follows from the definition of cosine similarity, the second step follows from $\|Y_{l,t}^{\Phi}\|_F = 1$ for any t, the third steps from Fact 6, the fourth step follows from Fact 8 and we expand $Y_{l,t}^{\Phi}$ around 0, the fifth step follows from the simple algebra, the sixth follows from the definition of inner product, the last step follows from we define $W_l^{\Phi} := (Y_{l,t-1}^{\Phi})^{\top} \cdot J$. □