Accelerate Diffusion Transformers with Feature Momentum

Anonymous Author(s)

Figure 1: A gallery comparing various acceleration methods and generation quality. We introduce FeMo, which observes the continuity of feature trajectory of diffusion models in different timesteps and further stabilizes it with an adaptive *momentum* mechanism, leading to 5.55× and 6.22× acceleration in FLUX and DiT without notable drop in generation quality.

Abstract

Diffusion models have demonstrated outstanding generative capabilities in image and video synthesis. However, their heavy computational burden, particularly due to the sequential denoising process and large model sizes, makes them challenging to meet real-time application demands. In this paper, motivated by the continuity of diffusion models in the feature space, we introduce FeMo, which employs a momentum mechanism to stabilize the dynamics of diffusion models in different timesteps, allowing us to accurately predict the features in the future timesteps based on the historical information. Additionally, we further propose an Adapted-FeMo, which allows for adaptive searching for the optimal coefficient for each generated sample. Extensive experiments demonstrate its effectiveness, *e.g.*, a **4.99× acceleration** on FLUX with **0.86% improvements** on image reward.Under the condition of maintaining generation quality, Adapted-FeMo achieves a maximum speedup of **7.10×** on DiT and **6.24×** on FLUX. Our codes are available in the supplementary material and will be released on Github.

CCS Concepts

• **Generative Multimedia → Multimedia Foundation Models**.

Keywords

Diffusion Models, Model Acceleration, Adaptive Momentum

1 Introduction

In the field of generative artificial intelligence, Diffusion Models (DMs) [8] have made significant progress, achieving excellent results in tasks such as image generation and video synthesis [1, 20]. Diffusion Transformers (DiT) [18] have further improved visual generation quality by replacing the U-Net architecture with the Transformer encoder architecture. However, these advancements come with a substantial increase in computational demands, and the high-order time complexity caused by the repeated computation

ACM MM, 2025, Dublin, Ireland

ACM ISBN 978-1-4503-XXXX-X/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

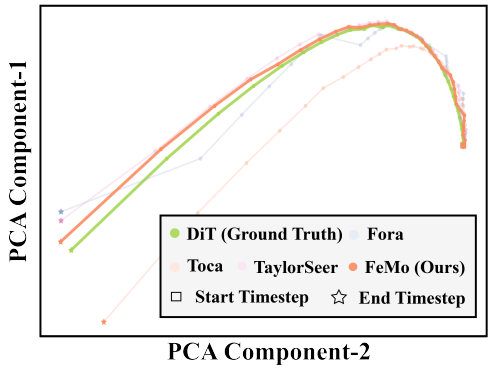

Figure 2: Scatter plot of the trajectories of FeMo and other baseline methods after performing Principal Component Analysis (PCA) on features.

of high-dimensional features during inference limits the feasibility of diffusion transformers in practical applications. To address the issue of computational inefficiency, several acceleration techniques have been proposed [16, 17, 28, 29].

Most recently, based on the observation that diffusion models exhibit strong similarity in features between adjacent timesteps, feature caching has been proposed as a plug-and-play technique to accelerate diffusion models [23]. Feature caching stores the features of diffusion models in previous *activated* timesteps and reuses them in the following *caching* timesteps, thus achieving significant acceleration by skipping the computation in the caching steps. Meanwhile, many studies have incorporated the characteristics of diffusion models, proposing methods such as caching important tokens [30], [31], and caching only the gap between features [2]. These works follow the "cache-then-reuse" paradigm that assumes that features in the previous timesteps are identical to the features in the following timesteps, which is approximately reasonable for temporally adjacent timesteps, but entirely invalid when applied to distant timesteps, leading to a significant generation quality degradation in high acceleration ratios.

Features of diffusion models are dynamic instead of static. More surprisingly, by visualizing the features of diffusion models in different timesteps, we find that it forms a relatively stable and continual trajectory. Based on this observation, [12] proposed TaylorSeer, a new "cache-then-forecast" paradigm that uses differential approximations of Taylor series expansions to predict the features at the current reuse step, providing a very preliminary solution to model the dynamics in the features of diffusion models. In this paper, we identify this paradigms suffer from the following issues.

First, Taylor-based approximations are inherently susceptible to *noise-sensitive gradient accumulation* during multi-step reuse, as high-order derivatives such as second- or third-order terms are prone to estimation inaccuracies that propagate exponentially across sequential steps. Furthermore, the *fixed-order truncation* of Taylor series fundamentally limits its capacity to model long-term dependencies, as predefined polynomial degrees fail to account for directional reversals in feature trajectories over extended horizons. Additionally, as shown in Figure 3, there is a huge difference in the feature trajectory across timesteps for different generated samples. However, previous methods ignore their difference and treat all of

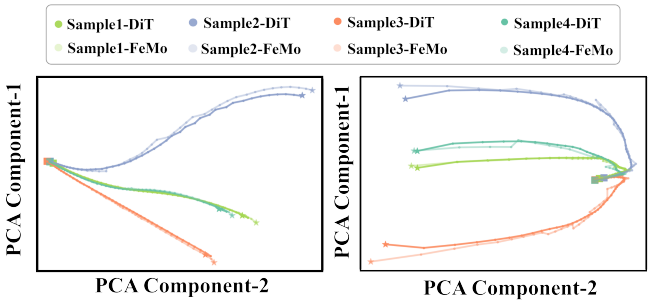

Figure 3: The trajectory of feature in diffusion models over timesteps for FeMoand the original DiT without acceleration on four different samples.

them with the same paradigm. These issues introduce the requirement for a *noise-robust, long-historical, and adaptive* technique to model the dynamics of the features in diffusion models.

To address these issues, we propose Feature Momentum (FeMo) to model the dynamics in the features of diffusion models by introducing an *adaptive momentum mechanism*, allowing us to predict the features in the future timesteps based on the trend of historical features. Concretely, FeMo employs a weighted prediction mechanism, utilizing the derivative terms approximated by the differences of all fully activated timesteps to predict the features at the current reused timestep. Building on this, Adapted-FeMo minimizes the discrepancy between predicted and actual features, dynamically adjusting the weights of historical features based on the feature distribution characteristics of different samples. Without the requirements on high-order derivatives, FeMo avoids the high sensitivity to the outlier in diffusion progress, allowing us to precisely match the original feature trajectory of the original diffusion models at a high acceleration ratio, as shown in Figure 2. Compared to previous caching methods where high reuse frequency led to severe image quality degradation, Adapted-FeMo is particularly effective when there is a large gap between fully activated steps. As shown in Figure 4, compared with previous SOTA methods, our approach reduces quality loss by 16 times, and still maintains good generative performance under an ultra-high acceleration ratio of up to 7.1×, whereas previous methods experienced significant generative failure at this acceleration ratio.

In summary, the main contributions of this work are as follows:

- We propose Feature Momentum (FeMo), which predicts features of diffusion models through an adaptive momentum mechanism. The accurate prediction in FeMo allows us to skip the computation in the future timesteps to achieve significant acceleration without drops in generation quality.
- Based on the difference in the feature trajectory of different diffusion models, we further propose Adapted-FeMo, which dynamically adjusts the weights of historical features for each generated sample to minimize the error caused by feature caching, thereby leading to superior generation performance.
- Extensive experiments on DiT and FLUX demonstrate that Adapted-FeMo achieves ultra-high speedups of **6.24**× and **7.10**× respectively, while maintaining nearly lossless generation quality, and sometimes even brings higher generation quality. It can be directly utilized in any diffusion transformer without requirements for additional training. Compared with TeaCache, FeMo achieves

a **29.34%** improvement in generation quality metrics at the highest acceleration ratio.

2 RELATED WORK

2.1 Diffusion Acceleration

Since the introduction of the Diffusion model [25], it has made significant progress in the field of generative models, thanks to its exceptional capabilities in generating images and videos. The initial model used the U-Net architecture [8, 20], but its computational cost and inference speed bottlenecks made it difficult to meet the needs of practical applications. Although later variants like DiT [18] enabled faster inference, they still required long generation times. To address this issue, various acceleration techniques have emerged in recent years , aiming to optimize the sampling process[14, 15, 26]and network structure [4, 5, 24]of Diffusion models, in order to improve their generation efficiency.

Reducing the number of sampling steps. DDIM [26] reduces the necessary sampling steps by introducing a non-Markov process, while maintaining high generation quality. In addition, some higher-order ODE (ordinary differential equation) solvers, such as the DPM-Solver series [14, 15], further accelerate the sampling process through more efficient numerical methods, reducing the computational load required for inference.

Optimizing the computational efficiency of denoising networks. For example, model compression techniques such as network pruning [4] and quantization [5, 24] can significantly accelerate inference speed without significantly reducing generation quality. Although these methods perform well on the U-Net architecture, there has been limited exploration of their application to Transformer architectures (e.g., Diffusion Transformer, DiT). Therefore, some new methods, such as FORA [23] and Δ-DiT [2], are specifically optimized for the characteristics of DiT.

2.2 Feature Cache

Feature caching technology has become an important direction for accelerating the inference of Diffusion models and has gained widespread attention in recent years. The core idea of this technology is to store and reuse intermediate features computed from previous steps during inference, in order to reduce redundant computations and improve computational efficiency. For example, DeepCache [16] and Faster Diffusion [10] cache feature maps from intermediate layers of the U-Net model and share computational results between adjacent steps, thereby significantly reducing the computational load during inference. These methods achieve acceleration without adding extra training burdens by reducing redundant computations. However, traditional feature caching methods are primarily designed for the U-Net architecture [8, 20] and are difficult to directly apply to Transformer-based Diffusion models [18]. Due to the differences in the self-attention mechanism and hierarchical structure of the Transformer architecture, traditional caching methods often fail to effectively reuse features, leading to a decline in generation quality. To address this, some new methods have proposed caching strategies specifically for Transformer architectures [16, 30, 31] and other related adaptive optimization algorithms [11, 12, 19, 28].

Learning to optimize caching strategies. The Learning-to-Cache method proposed by Ma et al. [16] improves caching efficiency by learning the optimal caching strategy, although this requires additional training steps. Additionally, ToCa [30] and DuCa [31] reduce information loss through dynamic selection feature updates.

Adaptive optimization strategies. At the same time, TeaCache [11] optimizes caching decisions by dynamically selecting time steps and estimating the differences between them. DiTFastAttn [28] reduces redundancy in self-attention computations across multiple dimensions by introducing windowed attention, feature similarity across time steps, and the elimination of conditional redundancy. EOC [19] presents an error optimization framework that enhances caching efficiency by leveraging prior knowledge extraction and adaptive optimization.Recently, [12] proposed refining the values in the cache by approximating the true values during the next sampling step using Taylor expansion terms.

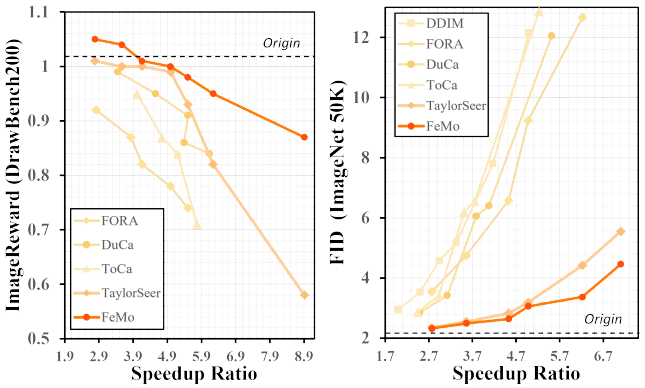

Figure 4: Comparison between the previous methods and FeMoon DrawBench with FLUX and ImageNet with DiT.

These innovative feature caching techniques provide new acceleration approaches for Diffusion models within the Transformer architecture. By reducing redundant computations, approximating true values in the cache, and adaptive optimization, they significantly improve inference speed while ensuring generation quality. However, these methods still face a challenge: as the time steps increase, the similarity between features rapidly decreases, leading to degradation in generation quality. Therefore, prediction-based caching methods have become a new development trend. For example, by predicting the features of future steps, instead of directly reusing past features. Our work achieves the approximation of the "true values" during reuse in the cache with minimal additional computational cost, thus maintaining high generation quality.

3 METHODOLOGY

In the Methodology section, we briefly introduce the Diffusion model and Transformer Architecture, followed by Feature Caching and prediction for the Diffusion model. Then, we present the prediction principle of the **FeMo** method and introduce the **Adapted-FeMo** method, which adaptively adjusts the momentum term coefficient β based on the difference between the current predicted value and the true computed value.

3.1 Preliminary

*Diffusion model and Transformer Architecture.*The diffusion model consists of a forward process and a reverse process. The forward

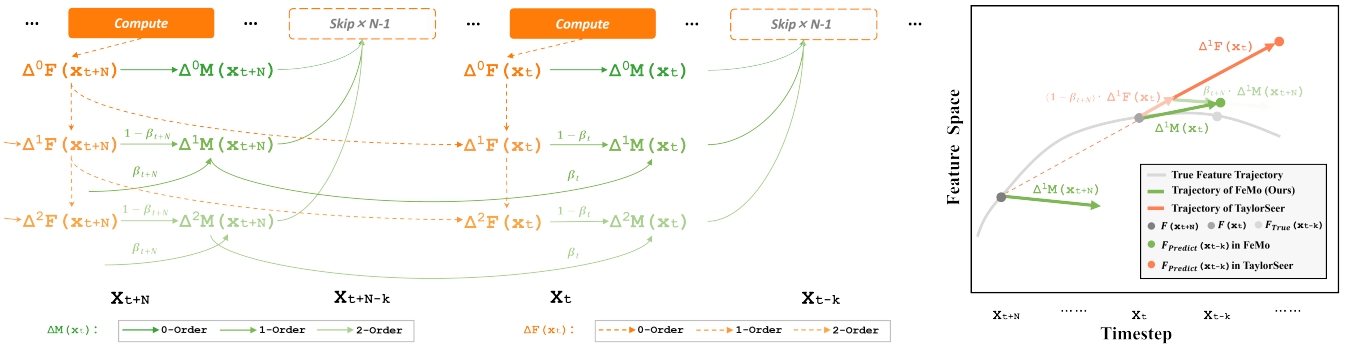

Figure 5: (a) FeMo (Order=2) uses first-order and second-order finite difference approximation derivatives, modeling and predicting the current reuse step feature by utilizing the historical information of each respective order. (b) From a conceptual perspective, the high-dimensional features are abstracted into a 2D space for vector analysis, illustrating that FeMo, when introducing the momentum term for prediction modeling, can make more accurate predictions in both direction and magnitude by considering the historical information of all previous full activation steps.

process gradually adds Gaussian noise to the clean image :

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon \tag{1}$$

while the reverse process gradually denoises the standard Gaussian noise to recover the real image. The denoising process is mainly based on calculating the posterior probability from the prior probability, which leads to the probability density function of the [noised] reverse process, as defined in the formula:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}\left(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right), \beta_t I\right) \tag{2}$$

In this process, T denotes the number of timesteps in the denoising process, $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$. ϵ_t represents a denoising network with inputs $\mathbf{x}_t$ and t. The training process involves optimizing θ such that the predicted noise removal approximates the noise added during the forward process. During image generation, the network ϵ_θ requires T inferences, consuming most of the computational cost in diffusion models. Recent studies suggest that replacing the traditional U-Net with a transformer-based architecture for ϵ_θ can significantly enhance generation quality. Diffusion Transformer models are usually composed of stacking groups of self-attention layers f_{SA}, multilayer perceptron f_{MLP}, and cross-attention layers f_{CA} (for conditional generation).The input data x_t consists of a sequence of tokens representing various patches within the generated images. This can be expressed as $x_t = \{x_i\}_{i=1}^{H\times W}$, where H and W correspond to the height and width of the images, or the latent code of the images, respectively.

*Feature Caching and predicting for Diffusion model.*Recent acceleration techniques apply Naïve Feature Caching in diffusion models by reusing features from adjacent timesteps:

$$\mathcal{F}(x_{t-k}) := \mathcal{F}(x_t), \quad \text{where } k \in \{1, \ldots, N-1\} \tag{3}$$

This strategy theoretically provides a speedup of $(N-1)$-times, but the errorr accumulation caused by direct reuse limits the maximum speedup before the model fails. A new method called TaylorSeer has recently been proposed. This method simulates the first-order derivative using finite differences, and applies Taylor expansion terms to make the historical features cached from the previous full computation approach the true feature values during the current reuse. The definition of the i-th forward finite difference is:

$$\Delta^i\mathcal{F}(x_t) = \Delta(\Delta^{i-1}\mathcal{F}(x_t)) = \Delta^{i-1}\mathcal{F}(x_{t+N}) - \Delta^{i-1}\mathcal{F}(x_t) \tag{4}$$

Where t is the current time step, and $t+N$ is the previous full computation step. And setting the base case $\Delta^0\mathcal{F}(x_t) = \mathcal{F}(x_t)$. Substituting Eq. 4 into the standard Taylor expansion, the general expression for approximating the reused features using the full computation step features is obtained:

$$\mathcal{F}(x_{t-k}) = \mathcal{F}(x_t) + \sum_{i=1}^{m} \frac{\Delta^i\mathcal{F}(x_t)}{i! \cdot N^i}(-k)^i \tag{5}$$

Although this method can effectively improve the accuracy under feature reuse, using only the expansion terms of the current full activation step as the direction guidance still lacks precision in determining the prediction direction in the vector space. This prediction can even have a negative effect at initial timesteps where feature changes are more significant. This still limits the model from achieving a larger speedup while maintaining generation quality. Therefore, we propose using the **historical gradient** to guide the prediction direction of the current timestep.

3.2 FeMo

*Feature prediction during the reuse step.*In order to suppress the oscillations caused by advancing in the direction of the finite difference approximation of the derivative, FeMo introduces a weighted historical momentum term based on the finite difference approximation. This helps smooth out short-term fluctuations in the prediction curve and further correct the predicted direction during the cache reuse step. The iterative formula for the momentum term that stores the historical difference terms is:

$$\mathcal{M}^i(x_t) = \beta \cdot \mathcal{M}^i(x_{t+N}) + (1-\beta) \cdot \nabla^i\mathcal{F}(x_t) \tag{6}$$

Here, β represents the weight of historical information in each iteration, and $(1-\beta)$ represents the weight of the differential derivative term calculated from the current full activation timestep.

The relationship for approximating the derivative using:

$$\Delta^i \mathcal{F}(x_t) \approx N^i \cdot \mathcal{F}^{(i)}(x_t) \tag{7}$$

By replacing the difference approximation term in Eq. 5 with the historical momentum term, and considering the proportional factor in Eq. 7, we derive the feature prediction formula for the k-th reuse timestep t using the m-th order derivative term:

$$\mathcal{F}(x_{t-k}) = \mathcal{M}(x_t) + \sum_{i=1}^{m} \frac{\mathcal{M}^i(x_t)}{i!}(-k)^i \tag{8}$$

Here, i is the order of differentiation, and $\mathcal{M}(x_t) = \mathcal{F}(x_t)$.

At this point, the feature prediction direction at timestep t during the cache reuse step is determined not only by the finite difference approximation of the derivative at the previous full activation step, but also by the predicted direction $\mathcal{M}^i(x_{t+N})$, which is obtained through a weighted vector average of all previously accumulated activation steps in the vector space. The direction of the optimization process tends to adjust gradually along the previous direction, reflecting the inertia effect and directional tendency in the optimization process.

Weighted Prediction Mechanism. As shown in Eq. 6, FeMo is a method that assigns different weights to all previous full computation steps, calculates the weighted moving average of local features, and uses the final moving average as the basis for determining the predicted feature values during the reuse step. To understand the weight of the influence of previous full computation steps on the current predicted features, and to make a reasonable initial setting, we derive the following general formula(We perform it when $m = 1$):

$$\begin{aligned}\mathcal{M}(x_t) = \beta^\tau \cdot \mathcal{M}(T) + (1-\beta) \cdot \left(\frac{\mathcal{F}(x_t)}{N} - \beta^{\tau-1} \cdot \frac{\mathcal{F}(x_{t+\tau N})}{N}\right) \\ - \sum_{j=1}^{\tau-1} \beta^{j-1} \cdot (1-\beta)^2 \cdot \frac{\mathcal{F}(x_{t+jN})}{N}\end{aligned} \tag{9}$$

Here, $t = T\%N$, $\tau = \frac{T-t}{N}$ and T is the full activation step closest to the first feature reuse step. It typically does not directly equal the total number of timesteps.

As observed from the Eq. 9, in the early stages of inference, the prediction direction and accuracy mainly depend on the computed values of $\mathcal{M}(T)$ and $\mathcal{F}(x_t)$. Due to the lack of sufficient historical data, the model's prediction error may be large. Given that generative models typically rely on limited prior information and data, we start saving the finite difference approximation values from the first timestep. This way, a larger initial value setting can provide stronger guidance in the early stages of the generation process, making it closer to the target distribution and reducing early fluctuations.

Bias Correction. At the same time, as observed in Figure 3 in the introduction, the feature trajectory is relatively smooth during the first few timesteps, with the finite difference derivative values being small (especially in higher-order approximations). To enhance the numerical stability of the FeMo method and ensures more accurate predictions, we have applied bias correction:

$$b = 1 - \beta^{\Delta t}, \mathcal{M}^i(x_t) = \frac{\mathcal{M}^i(x_t)}{b}, \tag{10}$$

where b is the bias correction term, and Δt represents the number of timesteps. When Δt is close to 0, the denominator can effectively amplify the current feature value, while when Δt is close to T, it has almost no impact on the current feature.

3.3 Adapted-FeMo

At the same time, we observe that when the feature trajectory is relatively smooth, assigning smaller weights to the historical gradients is sufficient for accurate predictions. However, in cases where the trajectory is not smooth—that is, when the values in the cache differ significantly from the true computed values—we can adaptively adjust the momentum term's weight based on the size of the local values. This allows us to effectively determine the prediction direction when significant changes occur in the feature's direction in the vector space. Therefore, we propose that FeMo perform an additional computation step during prediction, i.e., when t in Eq. 9 corresponds to the full computation step: (Mathematical analysis using $m = 1$ as an example).

$$\begin{aligned}\hat{y} &= \mathcal{M}^0(x_{t+N}) + N \cdot \mathcal{M}^1(x_{t+N}) \\ &= N[\cdot\beta^\tau \cdot \mathcal{M}(T) + (1-\beta) \cdot \mathcal{F}(x_t)] \\ &- N \cdot \beta^{\tau-1} \cdot \mathcal{F}(x_{t+\tau N}) + \mathcal{F}(x_{t+N}) \\ &- \sum_{j=1}^{\tau-1} \beta^{j-1} \cdot (1-\beta)^2 \cdot \mathcal{F}(x_{t+jN})\end{aligned} \tag{11}$$

At this point, we assume the objective is to minimize the mean squared error between the predicted value $formula\_value$(denoted as $\hat{y}$) and the computed value $true\_value$(denoted as y):

$$\min \quad \|y - \hat{y}\|_2 \tag{12}$$

At the same time, we solve the constraint function, so that at the current step t,all terms in the objective function, except for the variable β, are known tensors. Through first-order derivative analysis (Theorem A.2 in the appendix), we can deduce that β should satisfy:

$$\beta = \frac{(1-\tau) \cdot \mathcal{F}(x_{t+N})}{\tau \cdot N \cdot \mathcal{M}(T) - \mathcal{F}(x_{t+N})} \tag{13}$$

When $\|y\|_2 > \|\hat{y}\|_2$, it is clear that β should tend to increase and change relatively slowly. In order to achieve the optimization objective with the minimal computational effort, we set the learning rate γ and determine its specific value based on experimental experience. To implement the adaptive β, we use the following formula:

$$\beta_t = \beta_{t+N} + \mathrm{S} \cdot \gamma \tag{14}$$

When $\|\hat{y}\|_2 - \|y\|_2 < 0$, S is 1; otherwise, S is -1, if the difference is exactly 0, S is set to 0. At the same time, based on Eq. 13 and using a small sample experiment, we derive the initial value for β and restrict its range of variation.

Table 1: Quantitative comparison in text-to-image generation for FLUX on Image Reward.

Method FLUX.1[9]	Efficient Attention [3]	Acceleration				Image Reward ↑ DrawBench	CLIP ↑ Score
		Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑		
[dev]: 50 steps	✔	17.20	1.00×	3719.50	1.00×	0.9898	19.604
60% **steps**	✔	10.49	1.64×	2231.70	1.67×	0.9739	19.526
Δ-DiT ($\mathcal{N}$ = 2) †	✔	11.87	1.45×	2480.01	1.50×	0.9316	19.350
50% **steps** †	✔	8.80	1.95×	1859.75	2.00×	0.9429	19.325
40% **steps** †	✔	7.11	2.42×	1487.80	2.62×	0.9317	19.027
34% **steps** †	✔	6.09	2.82×	1264.63	3.13×	0.9346	18.904
Δ-DiT ($\mathcal{N}$ = 3) †	✔	8.81	1.95×	1686.76	2.21×	0.8561	18.833
FORA ($\mathcal{N}$ = 3) † [23]	✔	7.08	2.43×	1320.07	2.82×	0.9227	18.950
ToCa ($\mathcal{N}$ = 5) [30]	✘	10.80	1.59×	1126.76	3.30×	0.9731	19.030
DuCa ($\mathcal{N}$ = 5) [31]	✔	5.88	2.93×	1078.34	3.45×	0.9896	19.595
TaylorSeer ($\mathcal{N}$ = 4, $\mathcal{O}$ = 2) [12]	✔	6.81	2.53×	1042.28	3.57×	1.0024	19.402
Adapted-FeMo ($\mathcal{N}$ = 4, $\mathcal{O}$=1)	✔	6.67	2.58×	1042.28	**3.57×**	**1.0375**	**19.618**
FORA ($\mathcal{N}$ = 5) † [23]	✔	5.17	3.33×	893.54	4.16×	0.8235	18.280
TeaCache (l = 0.8) † [12]	✔	4.98	3.58×	892.35	4.17×	0.8683	18.500
ToCa ($\mathcal{N}$ = 8) † [30]	✘	8.47	2.03×	784.54	4.74×	0.9086	18.380
DuCa ($\mathcal{N}$ = 6) † [31]	✔	4.89	3.52×	816.55	4.56×	0.9470	19.082
TaylorSeer ($\mathcal{N}$ = 6, $\mathcal{O}$ = 2) [12]	✔	5.19	3.31×	744.81	**4.99×**	0.9953	**19.637**
Adapted-FeMo ($\mathcal{N}$ = 6, $\mathcal{O}$=1)	✔	5.07	3.39×	744.81	**4.99×**	**0.9984**	19.597
FORA ($\mathcal{N}$ = 7) † [23]	✔	4.22	4.08×	670.441	5.55×	0.7398	17.609
ToCa ($\mathcal{N}$ = 10) † [30]	✘	7.93	2.17×	714.66	5.20×	0.8390	18.165
DuCa ($\mathcal{N}$ = 9) † [31]	✔	7.28	2.36×	690.26	5.39×	0.8601	18.534
TaylorSeer ($\mathcal{N}$ = 7, $\mathcal{O}$ = 2)†[12]	✔	4.88	3.52×	670.44	**5.55×**	0.9331	19.553
TeaCache(l = 1.2) † [12]	✔	3.98	4.48×	669.27	5.56×	0.7351	18.080
Adapted-FeMo ($\mathcal{N}$ = 7, $\mathcal{O}$=1)	✔	4.70	3.66×	670.44	**5.55×**	**0.9770**	**19.556**
FORA ($\mathcal{N}$ = 9) † [23]	✔	4.42	3.90×	596.07	6.24×	0.5550	18.371
ToCa ($\mathcal{N}$ = 12) † [30]	✘	7.34	2.34×	644.70	5.77×	0.7131	17.907
DuCa ($\mathcal{N}$ = 10) † [31]	✔	6.5	2.65×	606.91	6.13×	0.8396	18.534
TeaCache (l = 1.4) † [12]	✔	3.63	4.91×	594.90	6.25×	0.7346	17.862
TaylorSeer ($\mathcal{N}$ = 8, $\mathcal{O}$ = 2) †[12]	✔	4.59	3.74×	596.07	**6.24×**	0.8167	19.499
Adapted-FeMo ($\mathcal{N}$ = 8, $\mathcal{O}$=1)	✔	4.37	3.94×	596.07	**6.24×**	**0.9501**	**19.550**

- † Methods exhibit significant degradation in Image Reward, leading to severe deterioration in image quality.

Figure 6: Qualitative comparison on FLUX.1-dev. Other methods encounter issues such as failure in text, wrong number of objects, low aesthetics, and so on, while FeMo achieves the best quality and acceleration.

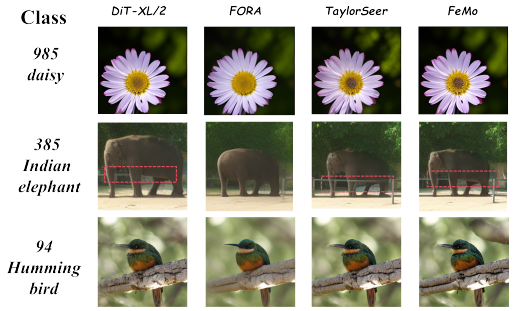

Figure 7: Detailed visualization results of different acceleration methods on DiT-XL/2 in the case of speed ratio of 6.22×.

4 EXPERIMENT

4.1 Experiment Settings

Model Configurations. The experiments are carried out using three advanced visual generative models: FLUX.1-dev[9], a text-to-image generation model ; and DiT-XL/2[18], a class-conditional image generation model. For more detailed model configurations, please refer to the Supplementary Material.

FLUX.1-dev[9] utilizes the Rectified Flow [13]sampling method with a standard configuration of 50 steps. All experimental evaluations were conducted on NVIDIA H20-NVLink GPUs.

DiT-XL/2 [18]adopts a 50-step DDIM[26] sampling strategy to ensure consistency with other models. Experiments on DiT-XL/2 were conducted on NVIDIA A800 80GB GPU

Evaluation and Metrics. In the text-to-image generation task, we performed inference on 200 prompts from **DrawBench** [22] to generate images with a resolution of 1000x1000, using **Image Reward** [27] and **CLIP score** [6] as the primary evaluation metrics. For the

Table 2: Quantitative comparison on class-to-image generation on ImageNet with DiT-XL/2.

Method DiT-XL/2[18]	Efficient Attention [3]	Acceleration				FID ↓	sFID ↓
		Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑		
DDIM-50 steps	✔	0.995	1.00 ×	23.74	1.00×	2.32	4.32
DDIM-25 steps	✔	0.537	1.85 ×	11.87	2.00×	2.95	4.51
Δ-DiT($\mathcal{N}$ = 2)	✔	0.322	3.09 ×	18.04	1.31×	2.69	4.67
Δ-DiT($\mathcal{N}$ = 3)	✔	0.301	3.31 ×	16.14	1.47×	3.75	5.70
DDIM-20 steps	✔	0.406	2.45 ×	9.49	2.50×	3.81	5.15
FORA ($\mathcal{N}$ = 3) [23]	✔	0.197	5.05 ×	8.58	2.77×	3.55	6.36
ToCa ($\mathcal{N}$ = 3) [30]	✘	0.216	4.78 ×	10.23	2.32×	2.87	4.76
DuCa ($\mathcal{N}$ = 3) [31]	✔	0.208	4.78 ×	9.58	2.48×	2.88	4.66
TaylorSeer ($\mathcal{N}$ = 3, $\mathcal{O}$ = 4) [12]	✔	0.292	3.41 ×	8.56	2.77×	2.35	4.69
Adapted-FeMo ($\mathcal{N}$ = 3, $\mathcal{O}$ = 2)	✔	0.241	4.13 ×	8.56	**2.77×**	**2.32**	**4.65**
DDIM-12 steps†	✔	0.228	4.36 ×	5.70	4.17×	7.80	8.03
FORA ($\mathcal{N}$ = 4) [23]	✔	0.169	5.89 ×	6.66	3.56×	4.30	7.37
ToCa ($\mathcal{N}$ = 6)†[30]	✘	0.163	6.10 ×	6.34	3.75×	6.55	7.10
DuCa ($\mathcal{N}$ = 6)† [31]	✔	0.127	7.83 ×	5.81	4.08×	6.40	6.71
TaylorSeer ($\mathcal{N}$ = 5, $\mathcal{O}$ = 4)[12]	✔	0.245	4.06 ×	5.24	4.53×	2.74	5.82
Adapted-FeMo (N = 5, O = 2)	✔	0.166	5.99 ×	5.24	**4.53×**	**2.64**	**5.30**
DDIM-10 steps†	✔	0.224	4.44 ×	4.75	5.00×	12.15	11.33
FORA ($\mathcal{N}$ = 7)† [23]	✔	0.142	7.01 ×	3.82	6.22×	12.55	18.63
ToCa ($\mathcal{N}$ = 13)† [30]	✘	0.146	6.82 ×	4.03	5.90×	21.24	19.93
DuCa ($\mathcal{N}$ = 12)† [31]	✔	0.131	7.60 ×	3.94	6.02×	31.97	27.26
TaylorSeer (N = 7, O = 4)[12]	✔	0.220	4.52 ×	3.82	6.22×	3.59	7.07
Adapted-FeMo ($\mathcal{N}$ = 7, $\mathcal{O}$ = 2)	✔	0.133	**7.48** ×	3.82	**6.22×**	**3.36**	**5.63**
DDIM-7 steps†	✔	0.168	5.92 ×	3.32	7.14×	33.65	27.15
FORA ($\mathcal{N}$ = 8)† [23]	✔	0.141	7.06 ×	3.34	7.10×	15.31	21.91
ToCa ($\mathcal{N}$ = 13)† [30]	✘	0.151	6.59 ×	3.66	6.48×	22.18	20.68
DuCa ($\mathcal{N}$ = 18)† [31]	✔	0.144	6.91 ×	3.59	6.61×	133.06	98.13
TaylorSeer ($\mathcal{N}$ = 9, $\mathcal{O}$ = 4)†[12]	✔	0.209	4.76 ×	3.34	7.10×	5.55	8.45
Adapted-FeMo (N = 9, O = 2)	✔	0.122	**8.16** ×	3.34	**7.10×**	**4.46**	**5.99**

- † Methods exhibit significant degradation in FID, leading to severe deterioration in image quality.

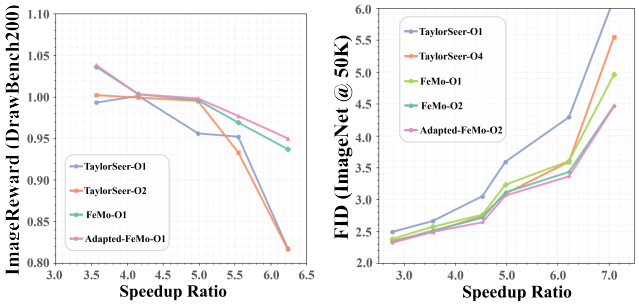

Figure 8: The ablation experiment results on FLUX.1-dev and DiT-XL/2 are visualized and compared with the TaylorSeer methods of various orders. The best-performing Adapted-FeMo shows significant improvements at high acceleration ratios and is able to maintain the generation quality without degradation at a high acceleration ratio of $\mathcal{N}$=9.

class-conditioned image generation task, we uniformly sampled from 1,000 **ImageNet** [21] categories and generated 50,000 images with a resolution of 256x256, using **FID-50k** [7] as the evaluation criterion, supplemented by **sFID** (Stabilized FID) for robustness evaluation. A detailed description can be found in the appendix.

4.2 Text-to-Image Generation

Quantitative Study. We compared Adapted-FeMo with existing methods. As shown in Table 1 ,although DuCa [31] ($\mathcal{N}$ = 5) achieves 3.45× FLOPs acceleration with an Image Reward of 0.9896, and ToCa [30] ($\mathcal{N}$ = 5) provides 3.30× acceleration, its image quality drops (0.9731). However, the performance of Adapted-FeMo ($\mathcal{N}$ = 4, $\mathcal{O}$ = 1) significantly outperforms both: with **3.57× acceleration**, it maintains an excellent Image Reward of **1.0375**. In comparison to the recent cache-based high-acceleration method TaylorSeer [12], which retains a stable Image Reward of 0.9331 at 5.55× acceleration, our Adapted-FeMo maintains an even better Image Reward (0.9770) and CLIP score (19.556) at the same **5.55× acceleration**. Notably, as the acceleration ratio increases, baseline methods suffer a significant degradation in image quality: ToCa ($\mathcal{N}$ = 12) drops to 0.7131 Image Reward at 5.77× acceleration, DuCa ($\mathcal{N}$ = 10) drops to 0.8396 Image Reward at 6.13× acceleration, and TaylorSeer ($\mathcal{N}$ = 8, $\mathcal{O}$ = 2) drops to 0.8167 Image Reward at 6.24× acceleration. In contrast, Adapted-FeMo ($\mathcal{N}$ = 8, $\mathcal{O}$ = 1) maintains an Image Reward of 0.9501 and a CLIP score of 19.550 even at **6.24× acceleration**, demonstrating an unparalleled balance of efficiency and fidelity.

Qualitative Study. Qualitative results in Figure 6 demonstrate that FeMo achieves outstanding generation quality while enabling high-speed inference. In the text generation task, such as *A sign that says 'Diffusion'*, FeMo accurately preserves the textual elements, whereas methods like **ToCa** and **DuCa** lose key details. In the generation task *Two cars on the street*, FeMo exhibits a strong ability to understand the prompt, while other methods show significant issues with color accuracy and quantity accuracy in the test cases. This indicates that FeMo strikes an excellent balance between speed and performance, especially in tasks that require fine detail preservation and a strong understanding of the prompt.

4.3 Class-Conditional Image Generation

Quantitative Study. We compared **Adapted-FeMo** with ToCa [30], FORA [23], DuCa [31], TaylorSeer [12], and methods that reduce DDIM steps on DiT-XL/2 [18]. The results show that Adapted-FeMo significantly outperforms other methods in terms of both acceleration ratio and image quality. As the acceleration ratio increases beyond 3.5×, the FID scores of methods like FORA, ToCa, and DuCa degrade significantly, leading to severe deterioration in image quality. In contrast, Adapted-FeMo maintains excellent generation quality even at **4.53× acceleration**, with a **FID of 2.68** and sFID of 5.30,superior to advanced baselines such as TaylorSeer, ToCa, and DuCa. Notably, Adapted-FeMo can still maintain good generation quality, without image degradation, even at the highest acceleration of **7.10**×, achieving an outstanding balance between efficiency and fidelity.

Qualitative Study. The qualitative results in Figure 7 demonstrate that FeMo successfully maintains the details and quality of the images during high-speed inference on the DiT-XL/2 model. In the generation task for the "*985 daisy*" class, FeMo accurately preserved the details of the flower. In the "*385 Indian elephant*" generation task, FeMo successfully modeled the relationship between the elephant's legs and the position of the fence, showing a good understanding of the physical spatial details and generation capabilities, in contrast to FORA, which failed to generate the outline, and TaylorSeer, which lacked modeling details. For the "*94 Humming bird*" class, FeMo demonstrated exceptional detail recovery ability, accurately representing the bird's feathers.

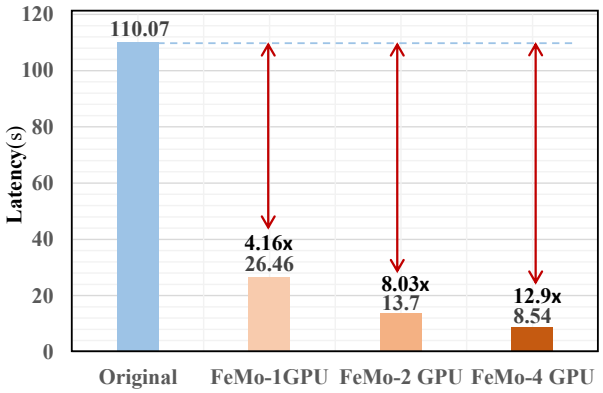

Figure 9: The latency behavior of FeMo under multi-GPU parallel acceleration with 2048×2048 **images on FLUX.**

5 DISCUSSION

5.1 Ablation Studies

We conducted ablation experiments on DiT-XL/2 [18] and FLUX.1-dev[9] to evaluate Adapted-FeMo and FeMo, focusing on the impact of the interval time parameter $\mathcal{N}$ and the order of the differential approximation $\mathcal{O}$ on computational efficiency and generation quality. The results show that, when using a first-order differential approximation ($\mathcal{O}$ = 1), Adapted-FeMo significantly outperforms the current state-of-the-art TaylorSeer on FLUX. On DiT, at a high acceleration ratio ($\mathcal{N}$ = 9), it also surpasses TaylorSeer in its full state ($\mathcal{O}$=4). Furthermore, when $\mathcal{O}$ = 2, Adapted-FeMo shows a noticeable improvement on DiT. At $\mathcal{N}$ = 7, it still achieves an FID of only 3.36 and can maintain image generation quality without degradation at $\mathcal{N}$ = 9. Meanwhile, the ablation results on FeMo show that our adaptive adjustment strategy continues to improve performance without affecting generation speed. The ablation experiments also demonstrate that using differential approximation derivative information from historical time steps to generate subsequent predictions effectively enhances prediction quality.

Overall, the Adapted-FeMo method demonstrates significant advantages in the current ablation experiments, especially in its performance at high acceleration ratios. Compared to existing methods, Adapted-FeMo achieves a higher acceleration ratio, a breakthrough that lays the foundation for its application in real-time or resource-constrained scenarios. *Detailed results can be found in the supplementary materials.*

5.2 The Stability of its Hyperparameters.

We chose to analyze the Adapted-FeMo ($\mathcal{N}$=9, $\mathcal{O}$=2) scheme on DiT-XL/2, where different γ values were analyzed within the same fluctuation range, as well as different ranges for the same γ, demonstrating the stability of hyperparameters within reasonable ranges, with maximum fluctuations of only 0.16% and 0.19%, respectively. More analysis can be found in the supplementary materials.

5.3 FeMo in Sequence Parallelism Technology.

As shown in the Figure 9, the proposed method is highly compatible with sequence parallelism technology. When generating images with a resolution of 2048, the latency on a single GPU is reduced from 2739.40 to 13.70, achieving a 1.93× speedup. On four GPUs in parallel, the latency is reduced from 1707.35 to 8.54, achieving a 3.10× speedup, indicating compatibility with parallel computation.

6 CONCLUSION

In this paper, to address the existing issues in the "cache-then-forecast" paradigm—where current methods are highly sensitive to gradient accumulation influenced by noise, struggle to handle long-term dependencies, and overlook the feature trajectory differences between different generated samples—we propose the FeMo method based on a weighted prediction mechanism. This method uses the differential approximation of derivatives from previously fully activated timesteps to predict the features at the current reuse step. Additionally, we introduce an adaptive mechanism that dynamically adjusts the weight of historical features during momentum updates based on each sample's feature trajectory characteristics.

References

[1] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, and Adam Letts. 2023. Stable video diffusion: Scaling latent video diffusion models to large datasets. In *arXiv preprint arXiv:2311.15127*.

[2] Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. 2024. -dit: A training-free acceleration method tailored for diffusion transformers. *arXiv preprint arXiv:2406.01125* (2024).

[3] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher R'e. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *arXiv preprint arXiv:2205.14135* (2022).

[4] Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2024. Structural pruning for diffusion models. In *Advances in Neural Information Processing Systems*, Vol. 36.

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

[6] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. 2022. CLIPScore: A Reference-free Evaluation Metric for Image Captioning. *arXiv preprint arXiv:2104.08718 [cs]* (2022).

[7] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2018. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *arXiv preprint arXiv:1706.08500 [cs]* (2018).

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, Vol. 33. 6840–6851.

[9] Black Forest Labs. 2024. Flux. https://github.com/black-forest-labs/flux

[10] Senmao Li, Taihang Hu, Fahad Shahbaz Khan, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, and Jian Yang. 2023. Faster diffusion: Rethinking the role of unet encoder in diffusion models. *arXiv preprint arXiv:2312.09608* (2023).

[11] Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. 2024. Timestep embedding tells: It's time to cache for video diffusion model. (2024).

[12] Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. 2025. Timestep Embedding Tells: It's Time to Cache for Video Diffusion Model. *arXiv* (March 2025). https://doi.org/10.48550/arXiv.2411.19108

[13] Xingchao Liu, Chengyue Gong, and et al. 2023. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*.

[14] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. 2022. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095* (2022).

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

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

[17] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. 2023. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14297–14306.

[18] William Peebles and Saining Xie. 2023. Scalable Diffusion Models with Transformers. *arXiv preprint arXiv:2212.09748 [cs]* (2023).

[19] Junxiang Qiu, Shuo Wang, Jinda Lu, Lin Liu, Houcheng Jiang, and Yanbin Hao. 2025. Accelerating diffusion transformer via error-optimized cache. (2025).

[20] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10684–10695.

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *arXiv preprint arXiv:1409.0575 [cs]* (2015).

[22] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. 2024. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. (2024).

[23] Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. 2024. Fora: Fast-forward caching in diffusion transformer acceleration. *arXiv preprint arXiv:2407.01425* (2024).

[24] Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. 2023. Post-training quantization on diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1972–1981.

[25] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*. PMLR, 2256–2265.

[26] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2021. Denoising diffusion implicit models. In *International Conference on Learning Representations*.

[27] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. 2023. ImageReward: Learning and Evaluating Human Preferences for Text-to-Image Generation. *arXiv preprint arXiv:2304.05977 [cs]* (2023).

[28] Zhihang Yuan, Hanling Zhang, Lu Pu, Xuefei Ning, Linfeng Zhang, Tianchen Zhao, Shengen Yan, Guohao Dai, and Yu Wang. 2024. DiTFastattn: Attention compression for diffusion transformer models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

[30] Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. 2024. Accelerating diffusion transformers with tokenwise feature caching. *arXiv preprint arXiv:2410.05317* (2024).

[31] Chang Zou, Evelyn Zhang, Runlin Guo, Haohang Xu, Conghui He, Xuming Hu, and Linfeng Zhang. 2024. Accelerating diffusion transformers with dual feature caching. (2024).