Rethinking Token-wise Feature Caching: Accelerating Diffusion Transformers with Dual Feature Caching

Chang Zou[1,*], **Shikang Zheng**[1,2,*], **Evelyn Zhang**[1], **Runlin Guo**[3], **Haohang Xu**[4], **Zhengyi Shi**[1]
Conghui He[5], **Xuming Hu**[6], **Linfeng Zhang**[1,†]
[1] Shanghai Jiao Tong University [2] South China University of Technology [3] Beihang University
[4] Huawei Technologies Ltd [5] Shanghai AI Lab [6] Hong Kong University of Science and Technology (Guangzhou)

arXiv:2412.18911v3 [cs.LG] 15 Aug 2026

I. Abstract

Diffusion Transformers (DiT) have become the dominant methods in image and video generation yet still suffer substantial computational costs. As an effective approach for DiT acceleration, feature caching methods are designed to cache the features of DiT in previous timesteps and reuse them in the next timesteps, allowing us to skip the computation in the next timesteps. Among them, token-wise feature caching has been introduced to perform different caching ratios for different tokens in DiTs, aiming to skip the computation for unimportant tokens while still computing the important ones. In this paper, we propose to carefully check the effectiveness in token-wise feature caching with the following two questions: (1) Is it really necessary to compute the so-called "important" tokens in each step? (2) Are so-called important tokens really important? Surprisingly, this paper gives some counter-intuition answers, demonstrating that consistently computing the selected "important tokens" in all steps is not necessary. The selection of the so-called "important tokens" is often ineffective, and even sometimes shows inferior performance than random selection. Based on these observations, this paper introduces dual feature caching referred to as DuCa, which performs aggressive caching strategy and conservative caching strategy iteratively and selects the tokens for computing randomly. Extensive experimental results demonstrate the effectiveness of our method in DiT, PixArt, FLUX, and OpenSora, demonstrating significant improvements than the previous token-wise feature caching.

***Index Terms*—Diffusion Acceleration, Feature Caching.**

II. Introduction

Diffusion Models (DMs) [1] have shown remarkable success in both image [2] and video generation [3]. Traditional DMs often use U-Net denoising structures, such as Stable Diffusion (SD) [2] and Stable Video Diffusion (SVD) [3], which have already demonstrated impressive results. Most recently, with the introduction of Diffusion Transformers (DiT) [4], the quality of visual generation has significantly improved further, propelling the development of numerous downstream applications [5]. However, the quality improvement is also accompanied by a significant increase in computational demands, reducing the inference efficiency of transformers and making their deployment in practical scenarios a pressing challenge.

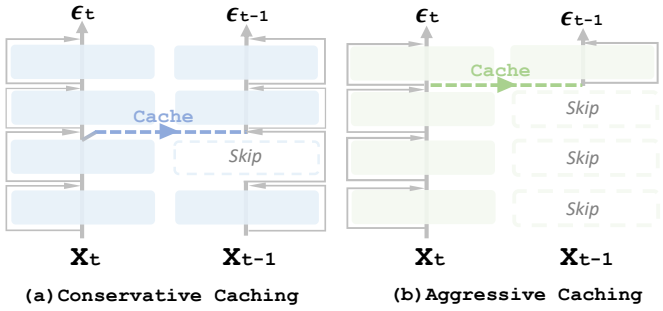


Fig. 1. Comparison between conservative caching (a) and aggressive caching (b): Conservative caching skips some tokens instead of all tokens in each layer to maintain the generation quality, but also leads to lower acceleration ratios. t and $t-1$ denote the refreshing and caching timesteps, respectively.

Recently, feature caching has become one of the most popular techniques for diffusion acceleration. Motivated by the high similarity between features in the adjacent timesteps, *aggressive* feature caching caches the features computed in a *freshing timestep*, and then reuse them in the following several *caching steps*, as shown in Figure 1(b). Such a fresh-then-cache strategy enables the diffusion model to save computational costs in the caching step and thus leads to significant acceleration. However, since the similarity of features in non-adjacent timesteps decreases quickly, the error introduced by aggressive feature caching can be accumulated at an exponential speed, leading to a significant drop in generation quality, and making aggressive feature caching impractical.

To solve this problem, recent work [6]–[9] introduces some computations during the caching steps in certain layers and tokens, which makes the cached features in previous steps corrected for the current caching step. Among them, token-wise feature caching (`ToCa`) has been proposed to selectively compute the important tokens while skipping the unimportant tokens with the cache, reducing the accumulated error from feature caching, but also making it difficult to achieve a higher acceleration ratio. In this paper, we rethink and distill `ToCa`. Specifically, we focus on the following two questions.

* Equal contribution.
† Denotes the corresponding author.

(I) Is it necessary to compute the important tokens in all caching steps? ToCa computes the important tokens in each caching step to reduce the accumulated error from feature caching. Figure 2 shows the caching error from the aggressive (marked in green) and conservative caching (ToCa, marked in blue). It is observed that the caching error from ToCa increases slowly as the caching step grows, while the error in the aggressive caching method accumulates rapidly with each caching step, which verifies the effectiveness of ToCa. However, we find that the caching error of the aggressive and conservative caching exhibit a close value at the first caching step (*i.e.*, timestep 27 in Figure 2), suggesting that computing the important tokens in ToCa is not always necessary. Concretely, when the error from caching has not been accumulated yet, computing important tokens does not bring benefits and can be removed for better efficiency.

Solution: Dual Feature Caching: Based on this observation, we propose DUal CAching, which leverages the advantages of aggressive and conservative caching methods. In each caching cycle, DuCa initializes the cache by performing the full computation in the *fresh* timestep. Then, it performs the one-step aggressive caching for high-ratio acceleration, followed by one-step conservative caching to fix the cache error. This alternative caching strategy can be repeated multiple times to achieve the best balance between efficiency and quality.

(II) Are so-called important tokens really important? The basic assumption of ToCa is that there exist some tokens that are sensitive or robust to feature caching. Based on this motivation, ToCa introduces four kinds of scores to describe the importance of each token based on their attention scores, spatial position, and frequency of being cached, respectively. Unfortunately, these scores not only introduce additional computation costs, but also lead to the incompatibility of FlashAttention, one of the default efficient operators in hardware. Hence, it results in around 100% increments in latency and improves the memory overhead from $O(N)$ to $O(N^2)$. To solve this problem, we carefully study the effectiveness of each score in ToCa. Surprisingly, as shown in Table I and will be discussed later, we find that selecting tokens with the highest attention scores in ToCa sometimes shows inferior performance than random token selection, indicating previous methods may misunderstand the importance of tokens.

Solution: Selecting tokens with low similarity: After carefully trying all kinds of importance scores for token selection, we find that choosing a subset of tokens that has the smallest similarity to other tokens is the only solution to beat random token selection. On the other hand, choosing a subset of tokens with maximal similarity to each other almost achieves the worst performance that we can have. Based on this observation, we argue that *token selection should focus on their duplication instead of their importance*. Thus, random selection can be a good choice since randomness guarantees choosing tokens with different semantic information and leads to no additional computation or memory overhead.

Extensive experiments have demonstrated the performance of our method in abundant benchmarks, including text-to-image generation with FLUX.1-dev, PixArt-α, text-to-video generation with OpenSora, and class-conditioned image generation with DiT. For instance, $3.45\times$ acceleration on FLUX.1-dev and $2.50\times$ acceleration on OpenSora with almost lossless quality. In summary, our main contributions are as follows:

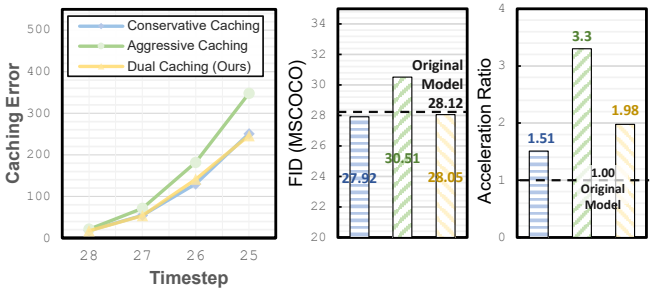


Fig. 2. The caching error and acceleration ratio of three feature caching strategies. The caching error is defined as the $L2$-norm distance between x_t computed with and without feature caching.

1) We rethink the necessity of computing "important" tokens in token-wise feature caching, and introduce DuCa, a dual feature caching strategy to perform the two caching methods alternatively, maintaining the generation quality while achieving a higher acceleration ratio.
2) We rethink the tokens selection methods in token-wise feature caching, finding that previous selection methods show inferior performance than random token selection, and explaining this phenomenon from the perspective of duplication. By just replacing previous methods with random selection, we achieve good compatibility with FlashAttention, leading to $\sim 2\times$ speed up in latency.
3) Extensive experiments have demonstrated DuCa in conditional image generation, text-to-image generation, and text-to-video generation, outperforming previous methods by a large margin. Our codes have been released.

III. Related Works

Acceleration of Diffusion Models To address the high computational costs commonly associated with DMs including DiT, various acceleration methods have been proposed. Existing acceleration methods can generally be divided into two categories: Designing sampling methods to reduce the number of steps needed to transition from noise to high-quality images, and directly accelerating the denoising model. Sampling-based acceleration methods enhance the efficiency of DMs by simply reducing the sampling steps. DDIM [10] leverages a deterministic, non-Markovian reverse process to significantly reduce generation steps while maintaining high-quality outputs. DPM Solver [11], through analytic derivation, optimizes the reverse process, enabling efficient sampling with fewer steps, while DPM Solver++ [12] further refines this approach by incorporating noise control and diverse sampling techniques to improve quality and adaptability. Rectified Flow [13] employs a recurrent flow framework, utilizing specially designed flow fields to accelerate the diffusion process.

Feature Caching Due to the high computational cost of retraining the diffusion models, or even small learnable modules introduce significant costs, limiting model reusability across different scales, resolutions, and tasks, driving interest in training-free acceleration methods. Faster Diffusion [2]

improves computational efficiency by caching the output of the encoder module on certain steps, while DeepCache [14] reduces redundant computations by reusing low-resolution feature information in the skip connections of the U-Net structure. Unfortunately, although both are training-free methods, they are specialized feature caching techniques designed specifically for U-Net-based models like Stable Diffusion [2], making them difficult to directly adapt to transformer-based models like DiT [6]. Recently, to address the lack of feature reuse solutions for DiT, Δ-DiT [6] and Pyramid Attention Broadcast (PAB) [15] have been introduced. The former focuses on constructing residuals of DiT outputs across different layers, while the latter emphasizes the importance of various types of attention within DiT Blocks containing multiple attention heads. Learning-to-Cache [7] achieves higher acceleration ratios by employing a learnable router to determine whether computations are needed at each layer, but it incurs prohibitive computational costs. These feature caching methods generally lack attention to finer-grained features. `ToCa` [9] proposes a method that selects important tokens at both layer and token granularity, allocating more computation to these tokens while caching less important ones, advancing temporal redundancy management to the token level.

IV. Method

A. Preliminary

a) Diffusion Models: Diffusion models [1] are designed with two main processes: a forward process, where Gaussian noise is incrementally added to a clean image, and a reverse process, where a standard Gaussian noise is gradually denoised to reconstruct the original image. Letting t represent the timestep and β_t the noise variance schedule, the conditional probability in the reverse (denoising) process, $p_\theta(x_{t-1} \mid x_t)$, can be modeled as:

$$\mathcal{N}\left(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right), \beta_t \mathbf{I}\right), \quad (1)$$

where $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$, and T represents the total number of timesteps. Notably, ϵ_θ is a denoising network parameterized by θ, which takes x_t and t as inputs to predict the noise needed for denoising. In an image generation procedure using $\mathcal{T}$ timesteps, ϵ_θ must perform inference $\mathcal{T}$ times, which constitutes the majority of computational costs in diffusion models. Recent studies have shown that implementing ϵ_θ as a transformer often improves the quality of generation.

b) Diffusion Transformers: Diffusion transformers (DiT) are usually composed of stacking groups of self-attention layers $\mathcal{F}_{\text{SA}}$, multilayer perceptron $\mathcal{F}_{\text{MLP}}$, and cross-attention layers $\mathcal{F}_{\text{CA}}$ (for conditional generation). It can be roughly formulated as:

$$\begin{aligned} \mathcal{G} &= g_1 \circ g_2 \circ \cdots \circ g_L, \quad \text{where} \\ g_l &= \mathcal{F}^l_{\text{SA}} \circ \mathcal{F}^l_{\text{CA}} \circ \mathcal{F}^l_{\text{MLP}}, \end{aligned} \quad (2)$$

$\mathcal{G}$, g_l, and $\mathcal{F}^l$ represent the DiT model, the blocks in DiT, and different layers in a DiT block, respectively. l denotes the index of DiT blocks, and L denotes the depth of DiT. In diffusion transformers, the input data $\mathbf{x}_t$ consists of a sequence of tokens representing different patches in the generated images, expressed as $\mathbf{x}_t = \{x_i\}_{i=1}^{H \times W}$, where H and W indicate the height and width of the images or their latent code, respectively. Then, the computation between $\mathcal{F}$ and $\mathbf{x}$ can be generally written as $\mathcal{F}(\mathbf{x}) = \mathbf{x} + \text{AdaLN} \circ f(\mathbf{x})$, where the first item $\mathbf{x}$ indicates the residual connection and f denotes the MLP or attention layers. AdaLN indicates the adaptive layer normalization [16].

Feature Caching: Feature caching aims to reuse the features that are computed and cached in the past timesteps to skip the computation in future timesteps. It is usually performed in multiple repeated caching cycles. Assuming a caching cycle with N timesteps from the timestep t to $t+N-1$, then feature caching performs the full computation in the timestep t and stores the obtained features in a cache. This progress can be denoted as $\mathcal{C}[l] := g_l(x_t)$, where $\mathcal{C}$ denotes the cache and ":=" denotes assigning value operation. Then, in timesteps from $t+1$ to $t+N-1$, the computation in these steps is skipped by reusing the cached features, which can be formulated as $g_l(x_{t+i}) := \mathcal{C}[l], i \leq N-1$. For simplicity, in this paper, the timestep t and the timesteps in $[t+1, t+N-1]$ are named the freshing step and the *caching* steps, respectively.

B. Dual Feature Caching

The proposed dual feature caching aims to perform aggressive caching and conservative caching alternatively in the caching steps. Hence, in this section, we first introduce the details of the two caching strategies.

a) Aggressive Caching in `DuCa`: As introduced in Figure 1(b), aggressive caching directly replaces the output of the DiT block with the cached feature which is computed in previous layers, which means that all computations of the first l layers can be skipped. This can be expressed as $g_l := \mathcal{C}[l]$, where $\mathcal{C}[l]$ denotes the cached feature at the l_{th} layer. Then, the computation with aggressive caching can be written as:

$$\mathcal{G}(x)_t := g_{l+1} \circ ... \circ g^t_L(\mathcal{C}[l]). \quad (3)$$

To achieve the maximal acceleration with aggressive caching, we set $l = L-1$ in most of our experiments, which indicates almost skipping all the layers in the caching step.

b) Conservative Caching in `DuCa`: As shown in Figure 1(a), conservative caching does not cache and reuse all the features. Instead, it caches and reuses only the features without the residual connection, which allows it to allocate the computation to some layers in the caching steps to correct the cached features. Recently, `ToCa` [9] introduced a token-wise approach that selectively reuses features based on the importance of each token and its sensitivity to caching, which enables us to perform computation only in the important tokens. Following `ToCa`, we first calculate an importance score $\mathcal{S}$ for each token x_i and then use this score to divide the index set of tokens $\mathcal{I}$ into two *complementary* sets: $\mathcal{I}_{\text{Cache}}$, for tokens to be cached, and $\mathcal{I}_{\text{Compute}}$, for tokens to be computed. Subsequently, the computation for the i_{th} token x_i on the l_{th} layer can be expressed as: $\gamma_i f^l(x_i) + (1-\gamma_i)\mathcal{C}[l](x_i)$, where $\gamma_i = 0$ for $i \in \mathcal{I}_{\text{Cache}}$ and $\gamma_i = 1$ for $i \in \mathcal{I}_{\text{Compute}}$. Here, $\mathcal{C}[l](x_i)$ denotes retrieving the cached value of x_i from the conservative cache $\mathcal{C}$ at layer l, which incurs no computation cost, enabling partial omission of calculations. After this selective computation in

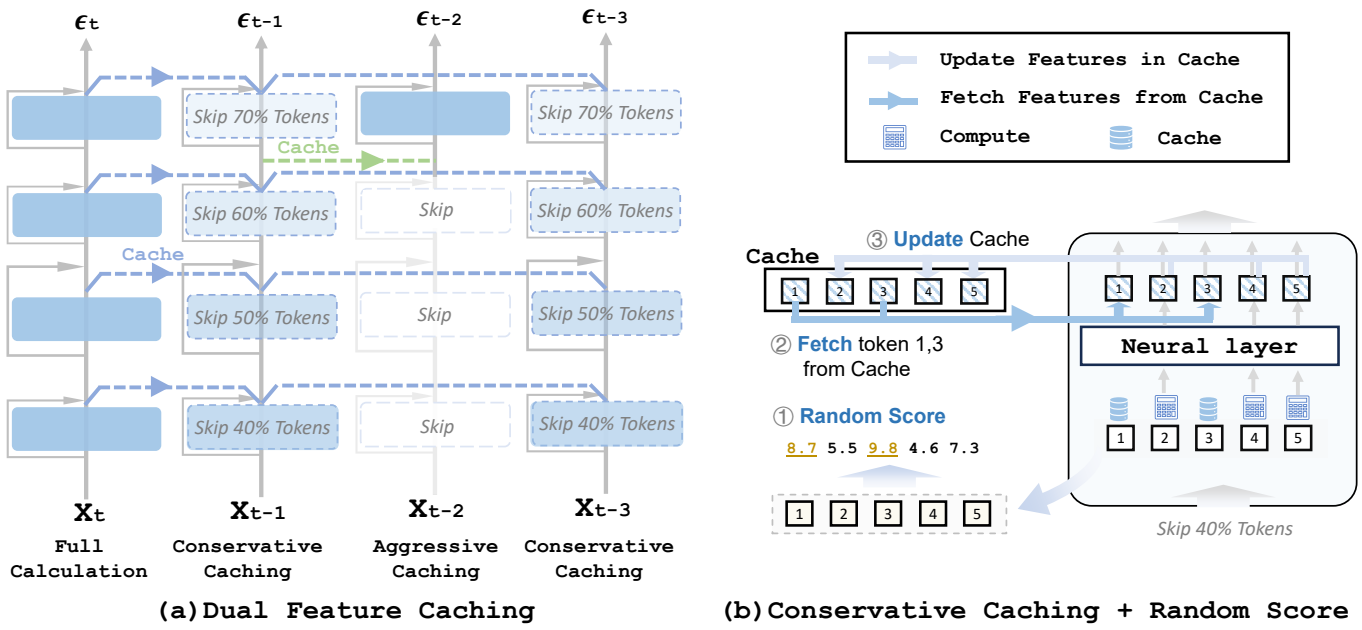


Fig. 3. The overview of DuCa. (a) In the first timestep of a cache cycle, DuCa computes all tokens and stores them in the cache as initialization. Then, DuCa performs a conservative caching step followed by an aggressive caching step, and then another conservative caching step, alternating in this manner to form a complete cache cycle. (b) The proposed method selects tokens for DuCa by setting random scores for all tokens to diversity. The tokens with the largest random scores are selected as cached tokens. The computation results for these tokens are directly retrieved from the cache, while the remaining tokens are computed through the neural layer, with their updated results stored back into the cache.

TABLE I
TOKEN SELECTION METHODS FOR FLUX ON IMAGE REWARD. THE EXPERIMENTS WERE CONDUCTED ON THE FLUX.1-DEV MODEL WITH A FULL ACTIVATION INTERVAL OF 4, SELECTING 10% OF TOKENS AT EACH CACHE STEP. A HIGHER IMAGE REWARD MEANS BETTER PERFORMANCE.

Score	Efficient Attention	Max♠	Min♡
Attention	✘	0.9798	0.9707
K-norm	✔	0.9783	0.9726
V-norm	✔	0.9747	0.9798
Similarity	✔	0.9715	0.9811
Random	✔	0.9806	
Original Model	✔	0.9898	

the token dimension, the computed features are updated to $\mathcal{C}[l]$ for future reusing.

c) Key Issue of Attention-Based Selection Methods: As the previous token-wise conservative caching method, ToCa offers two attention map-based selection strategies that have shown good effectiveness. However, the attention map-based selection strategy is incompatible with another commonly used attention-optimization method FlashAttention [17] and memory-efficient attention, which can not provide the attention scores but are capable of reducing the memory costs of attention from $O(N^2)$ to $O(N)$ and significant accelerating the attention layers. This conflict between attention scores-based token selection with this optimized attention computation makes ToCa less practical in application and raises the requirements for attention score-free token selection methods.

d) Diversity matters more in token selection: As shown in Table I, we compare attention scores with other classic token selection methods used in large language and multimodal models, such as K-norm and V-norm, which select tokens based on maximizing and minimizing the corresponding values, respectively. We then compare these results with the simplest and most naive random selection. Surprisingly, while the method of selecting tokens based on the highest attention score in ToCa is the best-performing approach among all standard-based selection methods, it still performs slightly worse than random selection.

The results suggest that rather than the traditional assumption that tokens can be distinguished as important or unimportant, selecting a diverse and broad range of tokens is more beneficial for performance. Based on this hypothesis, we designed an experiment that scores tokens based on similarity. For each cache step in Table I, 10% of all tokens were selected. First, 1% of all tokens were randomly chosen as the base tokens. Then, for the remaining 9%, tokens were selected based on two criteria: one method selects tokens with the highest similarity to the base tokens, aiming to choose tokens of the same type, while the other method selects those with the lowest similarity, encouraging greater diversity in the selected tokens. The method based on maximum similarity achieved the worst performance among all selection methods, while the method based on minimum similarity achieved the best performance, further confirming the *importance of diversity in token selection*. Considering that the difference between the minimum similarity-based method and the random selection method is small and it does not require additional computation, the random token selection method is recommended.

C. Overall Framework

Based on the understanding of aggressive and conservative caching, DuCa introduces a caching scheme that alternates conservative and aggressive caching in different steps. Random selection serves as the selection strategy for conservative

TABLE II
QUANTITATIVE COMPARISON IN TEXT-TO-IMAGE GENERATION FOR FLUX ON IMAGE REWARD.

Method FLUX.1 [18]	Efficient Attention [17]	Acceleration				Image Reward ↑ DrawBench	Geneval↑ Overall
		Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑		
[dev]: 50 steps	✔	17.20	1.00×	3719.50	1.00×	0.9898	0.6752
60% **steps**	✔	10.49	1.64×	2231.70	1.67×	0.9739	0.6700
50% **steps**	✔	8.80	1.95×	1859.75	2.00×	0.9429	0.6655
40% **steps**	✔	7.11	2.42×	1487.80	2.62×	0.9317	0.6606
Δ-DiT ($\mathcal{N}=2$)	✔	11.86	1.45×	2480.01	1.50×	0.9444	0.6658
Δ-DiT ($\mathcal{N}=3$)	✔	8.68	1.98×	1686.76	2.21×	0.8721	0.6413
TeaCache	✔	7.41	2.32×	1114.44	2.63×	0.9344	0.6722
FORA [8]	✔	7.08	2.43×	1320.07	2.82×	0.9227	0.6594
ToCa [9]	✘	10.80	1.59×	1126.76	3.30×	0.9731	0.6750
DuCa ($\mathcal{N}=4, R=90\%$)	✔	6.22	2.77×	1190.24	3.13×	<u>0.9895</u>	<u>0.6750</u>
DuCa ($\mathcal{N}=5, R=90\%$)	✔	<u>5.88</u>	<u>2.93×</u>	<u>1078.34</u>	<u>3.45×</u>	**0.9896**	0.6747
DuCa ($\mathcal{N}=6, R=90\%$)	✔	**5.14**	**3.35×**	**917.32**	**4.05×**	0.9654	**0.6756**

TABLE III
QUANTITATIVE RESULTS FOR HIDREAM-L1.

Method HiDream-l1	Acceleration				Image Reward ↑ DrawBench
	Latency(s) ↓	Speed ↑	FLOPs(T) ↓	Speed ↑	
[full]: 50 steps	48.81	×1.00	7780.0	×1.00	1.182
25 steps	24.41	×2.00	3890.0	×2.00	1.128
ToCa ($N=4$)	31.22	×1.56	2540.1	×3.06	1.111
PAB ($N=8$)	36.33	×1.34	5962.0	×1.30	0.8876
DuCa ($N=3, R=90\%$)	19.88	×2.46	2873.4	×2.71	1.152
DuCa ($N=4, R=90\%$)	15.36	×3.18	2197.3	×3.54	1.090

caching, as shown in Figure 3. This design leverages the realignment capability of token-wise conservative caching to correct the misaligned timestep information occurring in aggressive caching steps. Therefore, it enables the high local compression rate of aggressive caching to be applied to more steps while further reducing redundant computations in conservative caching steps. We arrange token-wise conservative caching steps on odd-numbered steps following a fresh step and aggressive caching steps on even-numbered steps.

V. EXPERIMENTS

A. Experiment Settings

1) Model Configurations: Experiments are conducted on four widely known DiT-based models on the three generation tasks with the correlating default samplers, including the FLUX.1-dev model [18] with 50 rectified flow sampling steps, HiDream model with 50 rectified flow sampling steps, OpenSora [19] with 30 rectified flow sampling steps [13] for text-to-video generation, and DiT-XL/2 [4] with 50 DDIM [10] steps for class-conditional image generation. All the experiments are conducted with NVIDIA H800 80GB GPUs. Additionally, we have supplemented our experiments with the smaller text-to-image generation model PixArt-α [20] with 20 DPM Solver++ [12] steps and included the detailed design of all experiments. More details are available in the appendix.

2) Evaluation and Metrics: **Text-to-image generation** For the FLUX.1-dev and HiDream model with a high resolution of 1000×1000, we utilized 200 high-quality prompts from DrawBench [21], which focus on generating various types of images. We then generated images using these prompts and employed Image Reward [22] to assess the quality of the generated images while using the GenEval [23] benchmarks to evaluate the alignment between the images and the text.

Text-to-video generation VBench [24] evaluation framework is applied to evaluate the performance of the text-to-video model. With 5 videos per prompt and 950 prompts from the framework, 4,750 480p, 9:16, and 2s videos are generated to evaluate the 16 aspects proposed by VBench.

Class-conditional image generation 50,000 256×256 images uniformly sampled from the 1,000 classes in ImageNet [25] are generated to evaluate the FID-50k [26]. Besides, sFID, Precision, and Recall are introduced as supplementary metrics.

3) Results on text-to-image generation: The generation results of the proposed DuCa under various configurations are compared against recent state-of-the-art acceleration methods, including ToCa [9], FORA [8], and a series of methods that directly reduce sampling steps. These comparisons are conducted on FLUX.1-dev [18] and HiDream at a resolution of 1000×1000, as detailed in Table II and III.

In terms of *generation quality*, the quantitative results on the Image Reward indicate that ToCa achieves a maximum computational compression (*i.e.*, FLOPs speedup) of 3.30×, with an Image Reward of 0.9731, representing a decrease of 0.0167 compared to the original model. In contrast, DuCa achieves an Image Reward score of 0.9896 under a computational compression of 3.45×, remaining almost unchanged compared to the original model, reducing the quality loss to 1.2% of that observed with ToCa.

From the perspective of text-image alignment, the quantitative results on the GenEval metric demonstrate that DuCa maintains nearly lossless performance compared to the original model. Even under a computational compression of up to 4.05×, DuCa still ensures excellent alignment capability. In contrast, FORA exhibits a noticeable decline in alignment ability at a compression ratio of 2.82×.

B. Quantitative and Qualitative Results

From a practical application perspective, although the computational compression ratio of the proposed DuCa at 3.45× is not significantly different from ToCa's 3.30×, DuCa not only further enhances generation quality but also achieves a latency speedup ratio of 2.93 due to its compatibility with FlashAttention [17]. This far surpasses ToCa's 1.59× speedup, making DuCa superior to the current state-of-the-art ToCa in both speed and generation quality.

TABLE IV
QUANTITATIVE COMPARISON ON CLASS-TO-IMAGE GENERATION ON IMAGENET WITH DIT-XL/2.

Method	Latency(s) ↓	FLOPs(T) ↓	Speed ↑	sFID ↓	FID ↓	Precision ↑	Recall ↑
DDIM-50 steps	0.417	23.74	1.00×	4.29	2.31	0.80	0.59
DDIM-25 steps	0.209	11.87	2.00×	4.63	3.07	0.79	0.58
DDIM-20 steps	0.167	9.49	2.50×	5.04	3.69	0.78	0.58
DDIM-17 steps	0.142	8.07	2.94×	5.65	4.48	0.77	0.56
FORA [8]	0.197	8.57	2.77×	6.32	3.76	0.79	0.54
Δ-DiT ($\mathcal{N}=2$)	0.246	18.04	1.31×	4.67	2.69	0.79	0.56
Δ-DiT ($\mathcal{N}=3$)	0.173	16.14	1.92×	5.7	3.75	0.79	0.54
ToCa(a)	0.195	8.73	2.72×	4.98	3.31	0.79	0.55
ToCa(b)	0.214	10.23	2.32×	4.73	2.88	0.79	0.57
DuCa(a)	**0.177**	**8.76**	2.71×	4.53	3.00	0.80	0.57
DuCa(b)	0.198	9.58	2.48×	**4.59**	**2.84**	**0.80**	**0.58**

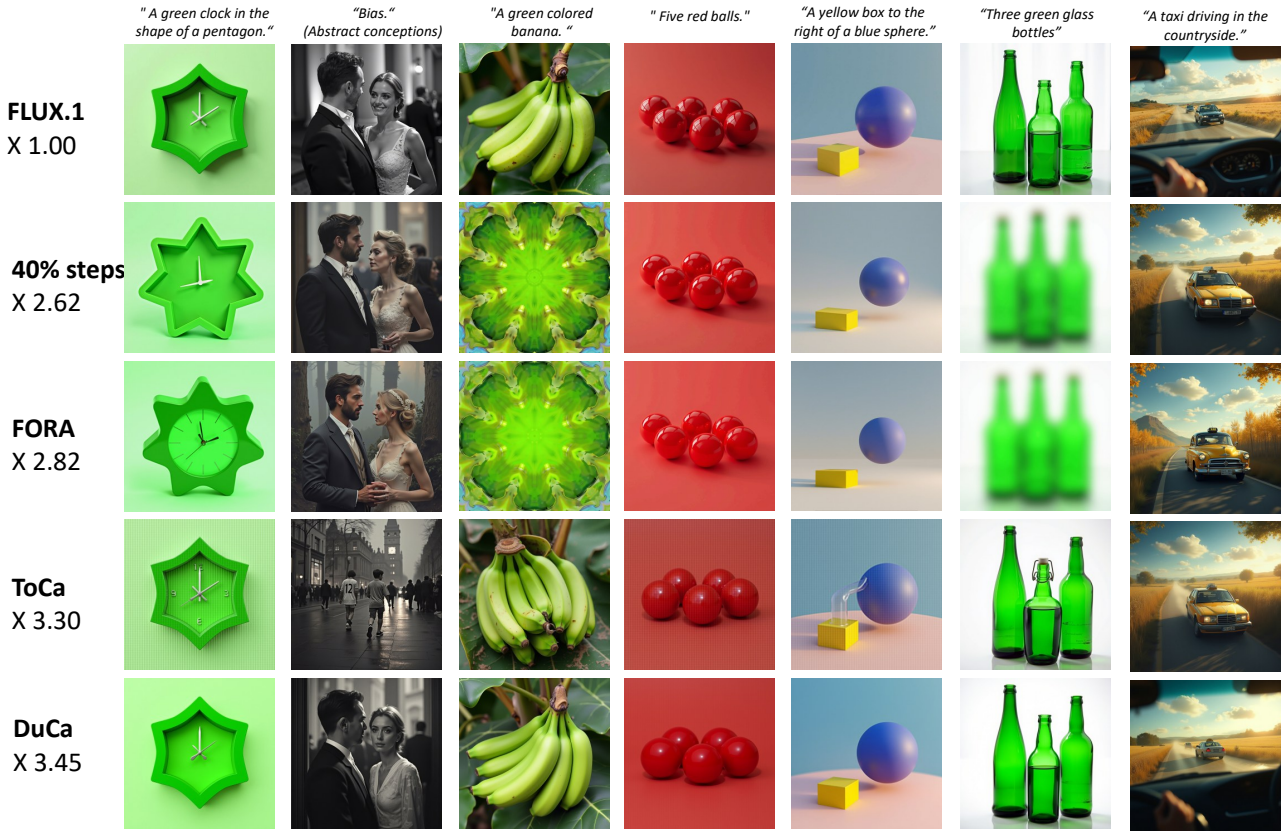


Fig. 4. Visualization results for different acceleration methods on FLUX.1-dev model.

In Figure 4, we further present the generation results of various acceleration methods. From Figure 4, it can be observed that across multiple samples, DuCa's generated images are closer to those of the original model without acceleration compared to other methods. The 40% steps and FORA schemes not only offer lower acceleration ratios but are also more prone to visual artifacts. Although ToCa exhibits better details to some extent, we notice that in the 1st, 4th, and 5th columns from left to right, there are inexplicable grid-like distortions. In contrast, DuCa emerges as a superior choice in terms of both speed and performance.

1) Results on text-to-video generation: The generation results of the state-of-the-art acceleration methods Δ-DiT, T-GATE [27], PAB [15], 15 rflow sampling steps, ToCa [9], FORA [8] and proposed DuCa on the text-to-video model OpenSora [19] are shown in Table V.

As shown in Table V, DuCa achieves a further acceleration from 2.36 × (as with ToCa) to 2.50 × with virtually no loss in generation quality. Additionally, despite having the lowest computational cost among all the compared methods, DuCa still achieves the highest VBench score of 78.83, demonstrating that the proposed strategy in DuCa, which alternates between aggressive and conservative caching steps, effectively eliminates the redundant computations in the fully conservative caching approach.

The advantages of DuCa, stemming from its diverse selection strategies, are also evident in its comparison with ToCa: although DuCa eliminates unnecessary redundant computations, resulting in lower actual computational load than ToCa, its VBench score decreases by only 0.58 compared to the

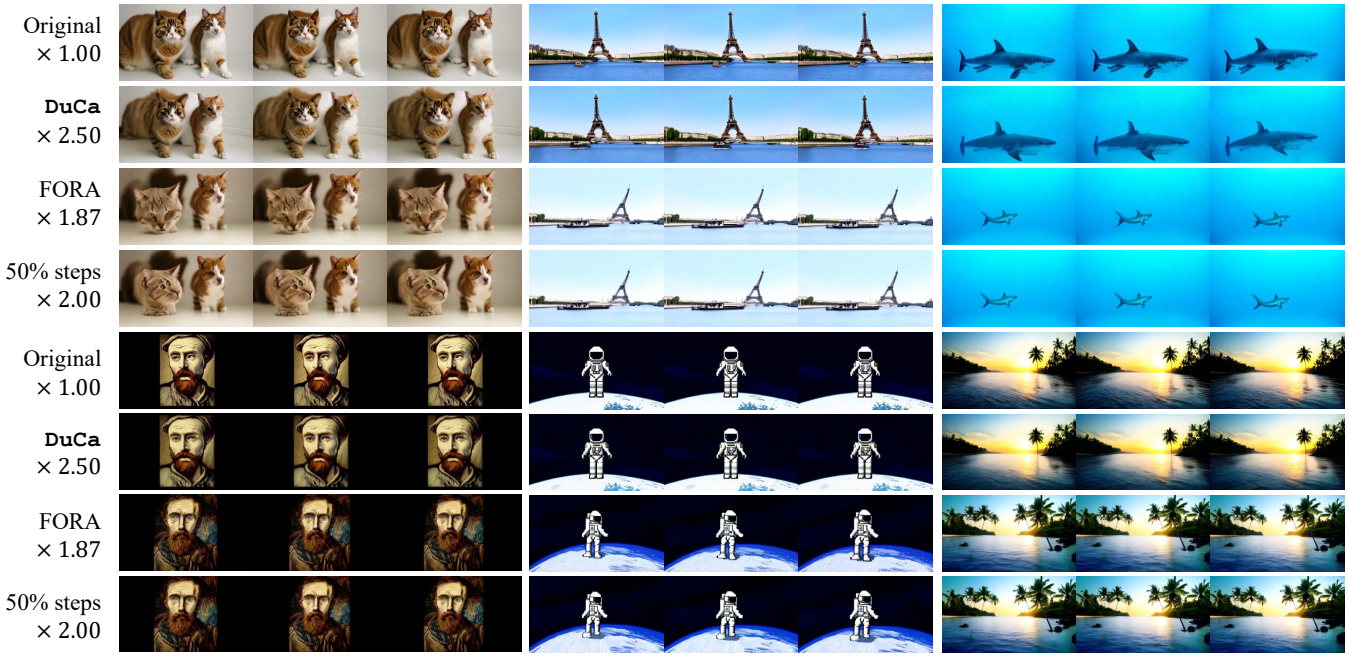


Fig. 5. Visualization results for different methods on OpenSora (videos available in the supplementary material).

TABLE V
QUANTITATIVE COMPARISON IN TEXT-TO-VIDEO GENERATION ON VBENCH. *RESULTS ARE FROM PAB [15]. PAB$^{1-3}$ INDICATE PAB WITH DIFFERENT HYPER-PARAMETERS.

Method	Latency(s)↓	FLOPs(T)↓	Speed↑	VBench(%)↑
OpenSora [19]	54.83	3283.20	1.00×	79.41
Δ-DiT* [6]	53.45	3166.47	1.04×	78.21
T-GATE* [27]	45.91	2818.40	1.16×	77.61
PAB^{1}* [15]	41.05	2657.70	1.24×	78.51
PAB^{2}* [15]	39.96	2615.15	1.26×	77.64
PAB^{3}* [15]	38.26	2558.25	1.28×	76.95
50% **steps**	28.85	1641.60	2.00×	76.78
FORA [8]	33.19	1751.32	1.87×	76.91
ToCa [9]	29.03	1394.03	2.36×	78.59
DuCa	**19.16**	**1315.62**	**2.50×**	**78.83**

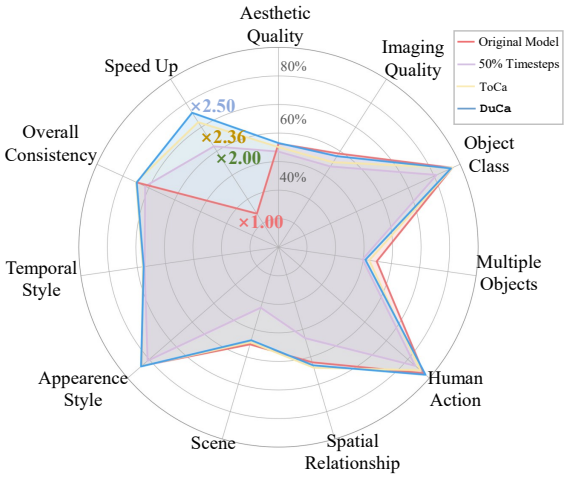


Fig. 6. VBench metrics and acceleration ratio of proposed **DuCa** and other methods.

original model, while ToCa's score drops by 0.82, indicating that DuCa reduces quality loss by 29.3%.

Furthermore, thanks to DuCa's compatibility with FlashAttention [17], the generation of a single 480p, 2s video takes only 19.16s with DuCa, compared to 29.03s with ToCa, resulting in a speedup of nearly 10s.

In Figure 6, we further present the performance of DuCa and the aforementioned methods across various dimensions of the VBench metric. DuCa shows results consistent with the non-accelerated original OpenSora model across multiple metrics. As shown in Figure 5, it is evident that DuCa is closer to the original model compared to other methods.

2) Results on class-conditional image generation: In Table IV, we compare ToCa [9], FORA [8], the direct reduction of the corresponding DDIM steps and the proposed DuCa. The experimental results indicate that with comparable FID scores, representing generation quality, DuCa further enhances acceleration. For instance, DuCa(b) with Self-Attention score achieves a similar FID-50k as ToCa(b), but the acceleration ratio increases from 2.32 to 2.48. Additionally, under a higher acceleration scenario with an acceleration ratio around 2.7, DuCa achieves an FID-50k of 3.00, further reducing the FID-50k by 0.31 compared to ToCa's 3.31, demonstrating that DuCa effectively reallocates computation.

TABLE VI
DIFFERENT CACHING METHODS ON FLUX.1-DEV.

Methods	Image Reward	FLOPs(T) ↓
Conservative-only	0.9789	1263.22
Aggressive-only	0.8539	1042.12
DuCa	**0.9895**	1190.24

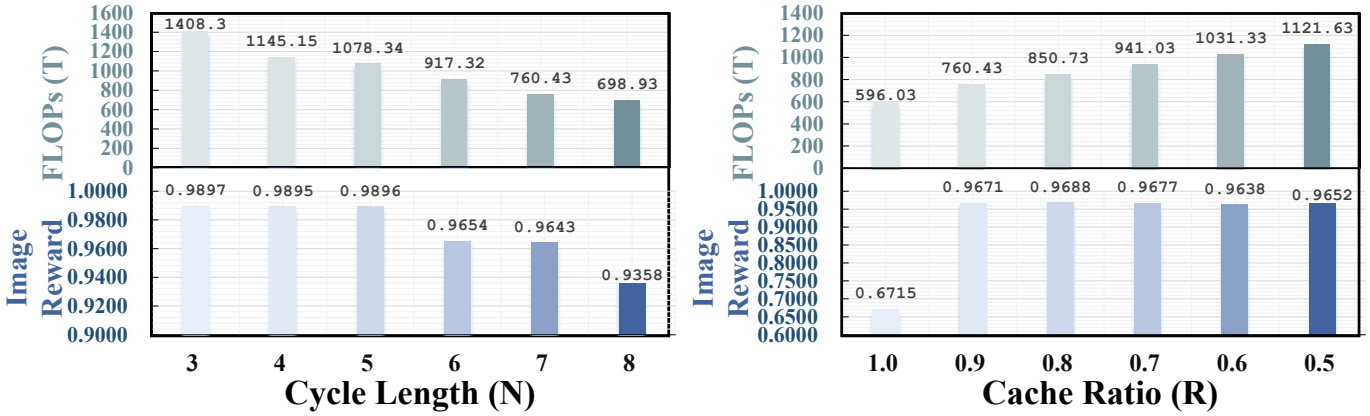


Fig. 7. Ablation study on cycle length N and cache ratio R. Setting $N = 5$ and $R = 0.9$ yields the best trade-off between computation and quality.

C. Ablation Studies

a) Diversity in Token Selection: As demonstrated in Table I, the results prove that the adopted random selection method, due to its consideration of diversity in token selection, outperforms a series of traditional methods based on specific metrics while maintaining compatibility with the Efficient Attention framework. Additionally, through experiments designed around similarity, we have explored and strongly validated the importance of diversity in token selection.

As shown in Table VI, we compare the conservative-only, aggressive-only, and alternating `DuCa` structures, where the aggressive structure is used in activated odd steps and the conservative structure in even steps, on FLUX.1-dev in terms of Image Reward and FLOPs. The conservative-only structure appears to have good generation quality but incurs an excessively high computational cost. In contrast, aggressive-only reduces computation to nearly half of conservative-only but suffers significantly in generation quality. Our proposed `DuCa` structure achieves a balanced computational load between the two, reducing the computational cost compared to the conservative-only method while maintaining comparable or even better generation quality.

b) Choice of Cycle Length and Cache Ratio: We further conduct ablation experiments to investigate the influence of cycle length (N) and cache ratio (R) on performance, as illustrated in Figure 7. When varying N from 3 to 8, we observe that shorter cycles incur higher FLOPs, while excessively long cycles result in noticeable quality degradation. The setting $N = 5$ provides a favorable trade-off, preserving generation quality while reducing computation. Similarly, for cache ratio R, lowering R below 0.7 significantly harms quality, whereas $R = 0.9$ eliminates redundant computation without compromising fidelity. Therefore, we adopt $N = 5$ and $R = 0.9$ as default hyperparameters in our main experiments.

VI. Conclusion

This paper first gives an in-depth study of the caching error on aggressive and conservative methods and then introduces dual caching `DuCa`, which performs aggressive and conservative alternatively to leverage their advantages in different caching steps. Moreover, we have discovered the value of random selection and further explored its implications, revealing that diversity plays a more critical role in token selection than previously thought. Random selection not only demonstrates excellent performance but also seamlessly integrates with efficient attention methods like FlashAttention, making feature caching a practical and viable approach in real-world applications. We hope this paper can attract more attention to feature caching for its real benefits, to make it simple but effective, instead of complex.

References

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

[2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

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

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

[5] K. Feng, Y. Ma, B. Wang, C. Qi, H. Chen, Q. Chen, and Z. Wang, "Dit4edit: Diffusion transformer for image editing," 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:273821065

[6] P. Chen, M. Shen, P. Ye, J. Cao, C. Tu, C.-S. Bouganis, Y. Zhao, and T. Chen, "δ-dit: A training-free acceleration method tailored for diffusion transformers," *arXiv preprint arXiv:2406.01125*, 2024.

[7] X. Ma, G. Fang, M. B. Mi, and X. Wang, "Learning-to-cache: Accelerating diffusion transformer via layer caching," *arXiv preprint arXiv:2406.01733*, 2024.

[8] P. Selvaraju, T. Ding, T. Chen, I. Zharkov, and L. Liang, "Fora: Fast-forward caching in diffusion transformer acceleration," *arXiv preprint arXiv:2407.01425*, 2024.

[9] C. Zou, X. Liu, T. Liu, S. Huang, and L. Zhang, "Accelerating diffusion transformers with token-wise feature caching," *arXiv preprint arXiv:2410.05317*, 2024.

[10] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *International Conference on Learning Representations*, 2021.

[11] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, "Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5775–5787, 2022.

[12] ——, "Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models," *arXiv preprint arXiv:2211.01095*, 2022.

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

[14] X. Ma, G. Fang, and X. Wang, "Deepcache: Accelerating diffusion models for free," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 15 762–15 772.

[15] X. Zhao, X. Jin, K. Wang, and Y. You, "Real-time video generation with pyramid attention broadcast," *arXiv preprint arXiv:2408.12588*, 2024.

[16] Y. Guo, C. Wang, S. X. Yu, F. McKenna, and K. H. Law, "Adaln: A vision transformer for multidomain learning and predisaster building information extraction from images," *J. Comput. Civ. Eng.*, vol. 36, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:250298381

[17] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. R'e, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *ArXiv*, vol. abs/2205.14135, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:249151871

[18] B. F. Labs, "Flux," https://github.com/black-forest-labs/flux, 2024.

[19] Z. Zheng, X. Peng, T. Yang, C. Shen, S. Li, H. Liu, Y. Zhou, T. Li, and Y. You, "Open-sora: Democratizing efficient video production for all," March 2024. [Online]. Available: https://github.com/hpcaitech/Open-Sora

[20] J. Chen, J. Yu, C. Ge, L. Yao, E. Xie, Y. Wu, Z. Wang, J. Kwok, P. Luo, H. Lu, and Z. Li, "Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis," in *International Conference on Learning Representations*, 2024.

[21] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi, "Photorealistic text-to-image diffusion models with deep language understanding," *ArXiv*, vol. abs/2205.11487, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:248986576

[22] J. Xu, X. Liu, Y. Wu, Y. Tong, Q. Li, M. Ding, J. Tang, and Y. Dong, "ImageReward: Learning and Evaluating Human Preferences for Text-to-Image Generation," Dec. 2023, arXiv:2304.05977 [cs]. [Online]. Available: http://arxiv.org/abs/2304.05977

[23] D. Ghosh, H. Hajishirzi, and L. Schmidt, "Geneval: An object-focused framework for evaluating text-to-image alignment," *ArXiv*, vol. abs/2310.11513, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:264288728

[24] Z. Huang, Y. He, J. Yu, F. Zhang, C. Si, Y. Jiang, Y. Zhang, T. Wu, Q. Jin, N. Chanpaisit *et al.*, "Vbench: Comprehensive benchmark suite for video generative models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 807–21 818.

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

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

[27] W. Zhang, H. Liu, J. Xie, F. Faccio, M. Z. Shou, and J. Schmidhuber, "Cross-attention makes inference cumbersome in text-to-image diffusion models," *arXiv preprint arXiv:2404.02747v1*, 2024.