

Timestep Embedding Tells: It’s Time to Cache for Video Diffusion Model

Feng Liu^{1*} Shiwei Zhang^{2†} Xiaofeng Wang^{1,3} Yujie Wei⁴ Haonan Qiu⁵
Yuzhong Zhao¹ Yingya Zhang² Qixiang Ye¹ Fang Wan^{1‡}

¹University of Chinese Academy of Sciences ²Alibaba Group

³Institute of Automation, Chinese Academy of Sciences

⁴Fudan University ⁵Nanyang Technological University

Project Page: <https://liewfeng.github.io/TeaCache>

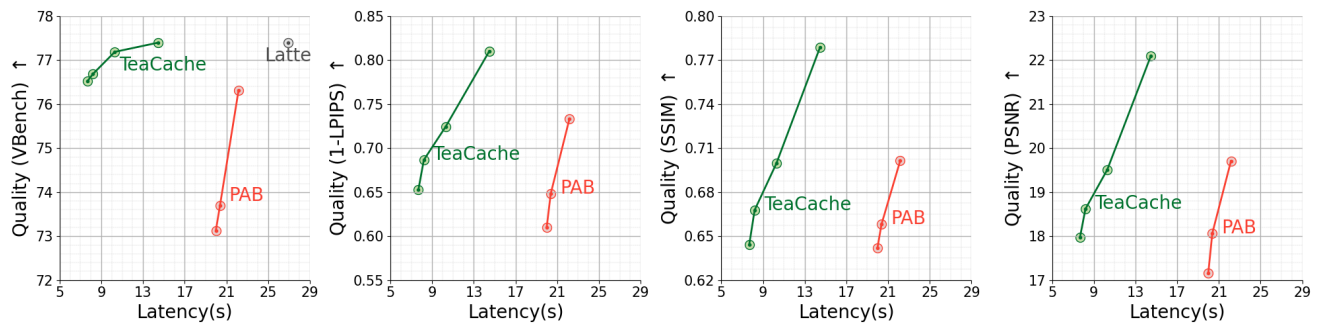


Figure 1. **Quality-latency comparison of video diffusion models.** Visual quality versus latency curves of the proposed TeaCache approach and PAB [61] using Latte [31]. TeaCache significantly outperforms PAB in both visual quality and efficiency. Latency is evaluated on a single A800 GPU for 16-frame video generation under 512×512 resolution.

Abstract

As a fundamental backbone for video generation, diffusion models are challenged by low inference speed due to the sequential nature of denoising. Previous methods speed up the models by caching and reusing model outputs at uniformly selected timesteps. However, such a strategy neglects the fact that differences among model outputs are not uniform across timesteps, which hinders selecting the appropriate model outputs to cache, leading to a poor balance between inference efficiency and visual quality. In this study, we introduce *Timestep Embedding Aware Cache (TeaCache)*, a training-free caching approach that estimates and leverages the fluctuating differences among model outputs across timesteps. Rather than directly using the time-consuming model outputs, *TeaCache* focuses on model inputs, which have a strong correlation with the model outputs while incurring negligible computational cost. *TeaCache* first modulates the noisy inputs using the timestep embeddings to ensure their differences better approximating those of model outputs. *TeaCache* then introduces a rescaling strategy to

refine the estimated differences and utilizes them to indicate output caching. Experiments show that *TeaCache* achieves up to $4.41 \times$ acceleration over *Open-Sora-Plan* with negligible (-0.07% Vbench score) degradation of visual quality.

1. Introduction

Recent years have witnessed the emergence of diffusion models [13, 16, 46, 48], as a fundamental backbone for visual generation. The model architecture has evolved from U-Net [4, 35, 37] to diffusion transformers (DiT) [34], which greatly increased model capacities. Empowered by DiT, video generation models [1, 2, 22, 31, 58, 62] have reached a groundbreaking level.

Despite of the substantial efficacy of these powerful models, their inference speed remains a pivotal impediment to wider adoption [25]. This core limitation arises from the sequential denoising procedure inherent to their reverse phase, which inhibits parallel decoding [42]. Moreover, as model parameters scale up and the requirements for higher resolution and longer durations of videos escalate [9, 58], the inference process experiences a further decline in speed.

To accelerate the visual generation procedure, distilla-

*Work was done during internship at Alibaba Group.

†Project Leader. ‡Corresponding author.

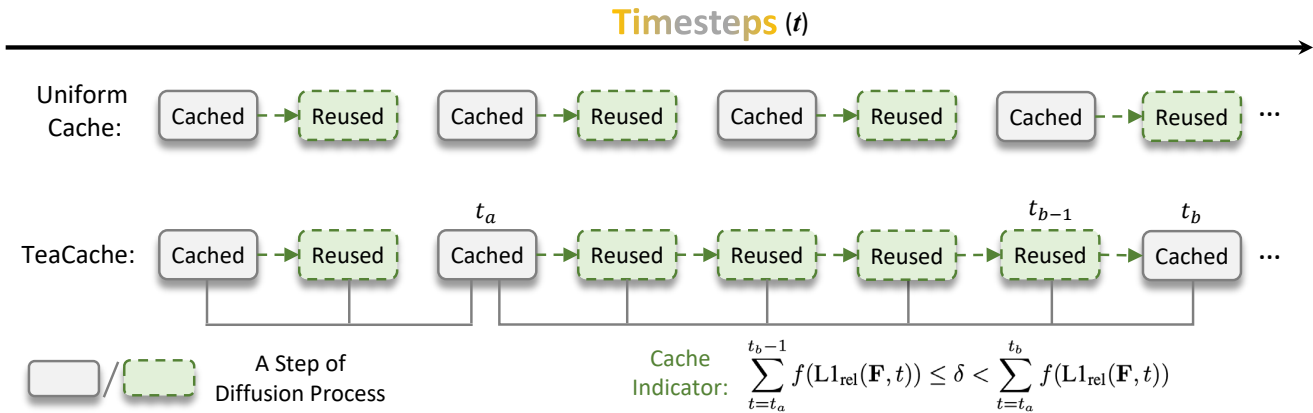


Figure 2. Comparison of the proposed TeaCache and the conventional uniform caching strategy for DiT models during inference. TeaCache is capable of selectively caching informative intermediate model outputs during the inference process, and therefore accelerates the DiT models while maintaining its performance. \mathbf{F} and t respectively denote the model inputs of noisy input and timestep embedding. $L1_{rel}$ and f are difference estimation functions of model inputs. δ is an indicator threshold of whether to cache a model output or not.

tion [32, 39, 53] and post-training [10, 30] are employed. However, these methods typically require extra training, which implies substantial computational cost and data resources. An alternative technological pathway is to leverage the caching mechanism [3, 14, 43], which does not require additional training to maintain the performance of diffusion models. These methods [11, 40, 57, 61] find that the model outputs are similar between the consecutive timesteps when denoising and propose to reduce redundancy by caching model outputs in a uniform way, Fig. 2(upper). Nevertheless, when the output difference between consecutive timesteps varies, the uniform caching strategy lacks flexibility to maximize the cache utilization.

In this study, we aim to develop a novel caching approach by fully utilizing the fluctuating differences among outputs of the diffusion model across timesteps. The primary challenge is: when can we reuse cached output to substitute the current timestep’s output? Intuitively, this is possible when the current output is similar with the cached output, Fig. 2(upper). Unfortunately, such difference is not predictable before the current output is computed. Consequently, without the guidance of difference, the uniformly cached outputs becomes redundant and the inference efficiency remains low.

To conquer this challenge, we propose Timestep Embedding Aware Cache (TeaCache), a training-free caching strategy. TeaCache leverages the following prior: **There exists a strong correlation between a model’s inputs and outputs.** If a transformation relationship can be established between the input and output difference, one can utilize the difference among inputs as an indicator of whether the corresponding outputs need to be cached, Fig. 2(lower). Since inputs are readily accessible, this approach would significantly reduce computation cost. We then delve into the in-

puts of diffusion models: a noisy input, a timestep embedding, and a text embedding. The text embedding remains constant throughout the denoising process and cannot be used to measure the difference of input across timesteps. As for the timestep embedding, it changes as timesteps progress but is independent of the noisy input and text embedding, making it difficult to fully reflect the input information. The noisy input, on the other hand, is gradually updated during the denoising process and contains information from the text embedding, but it is not sensitive to timesteps. To accurately describe the model inputs and ensure their strong correlation with the outputs, TeaCache follows the inference process of diffusion and employ the timestep-embedding modulated noisy input as the final input embeddings, among which the difference are then used to estimated the output difference.

It is noteworthy that the input difference estimated above still exhibits a scaling bias relative to the output difference, which has been observed through empirical studies. That is because this strategy only captures the correlation trend between input difference and output difference. Considering that both input and output differences are already scalars, TeaCache further introduces a simple polynomial fitting procedure to estimate the scaling factors between them. With the correction of the scaling factors, the input difference can accurately reflect the output difference and is ultimately used as an indicator of whether the outputs need to be cached, Fig. 2(lower).

The contributions of this paper include:

- We propose TeaCache, a training-free approach which is completely compatible with DiT diffusion models, to estimate the difference of model outputs, selectively cache model outputs and speed up the inference process.
- We propose a simple-yet-effective two-stage strategy to

estimate the difference of model output through model input. The proposed strategy uses timestep-embedding modulated noisy input to perform coarse estimation and a polynomial fitting procedure for refinement.

- TeaCache speeds up SOTA generation models, OpenSora [62], OpenSora-Plan [22], and Latte [31], (PAB [61]) with large margins at negligible quality cost, Fig. 1.

2. Related Work

2.1. Diffusion Model

In the realm of generative models, diffusion models [16, 46] have become foundational due to their exceptional ability to produce high-quality and diverse outputs. Initially developed with the U-Net architecture, these models have demonstrated impressive performance in image and video generation [6, 7, 17, 35–37, 52, 54, 55].

However, the scalability of U-Net-based diffusion models is inherently constrained, posing challenges for applications requiring larger model capacities for enhanced performance. To address this limitation, Diffusion transformers (DiT) [34] represent a significant advancement. By utilizing the scalable architecture of transformers [50], DiT provides an effective means to increase model capacity. A notable achievement in this field is the advancement in generating long videos through the large-scale training of Sora [33], which employs a transformer-based Diffusion architecture for comprehensive simulations of the physical world. This underscores the considerable impact of scaling transformer-based Diffusion models. An increasing number of studies have adopted the Diffusion transformer as the noise estimation network [8, 9, 22, 31, 58, 62].

2.2. Diffusion Model Acceleration

Despite the notable performance of Diffusion models in image and video synthesis, their significant inference costs hinder practical applications. Efforts to accelerate Diffusion model inference fall into two primary categories. First, techniques such as DDIM [47] allow for fewer sampling steps without sacrificing quality. Additional research has focused on efficient ODE or SDE solvers [19, 21, 27, 28, 48], using pseudo numerical methods for faster sampling. Second, approaches include distillation [38, 53], quantization [15, 26, 41, 45], and distributed inference [23] are employed to reduce the workload and inference time.

However, these methods often demand additional resources for fine-tuning or optimization. Some training-free approaches [5, 51] streamline the sampling process by reducing input tokens, thereby eliminating redundancy in image synthesis. Other methods reuse intermediate features between successive timesteps to avoid redundant computations [44, 56, 60]. DeepCache [57] and Faster Diffu-

sion [24] utilize feature caching to modify the UNet Diffusion, thus enhancing acceleration. FORA [40] and Δ -DiT [11] adapts this mechanism to DiT by caching residuals between attention layers. PAB [61] caches and broadcasts intermediate features at various timestep intervals based on different attention block characteristics for video synthesis. AdaCache [20] proposes to dynamically adjust feature caching strategies based on content complexity. FasterCache [29] observes significant redundancy in CFG and optimizes the reuse of conditional and unconditional outputs. While these methods have improved Diffusion efficiency, enhancements for DiT in visual synthesis remain limited.

3. Methodology

3.1. Preliminaries

Denosing Diffusion Models. Diffusion models simulate visual generation through a sequence of iterative denoising steps. The core idea is to start with random noise and progressively refine it until it approximates a sample from the target distribution. During the forward diffusion process, Gaussian noise is incrementally added over T steps to a data point \mathbf{x}_0 sampled from the real distribution $q(\mathbf{x})$:

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1 - \alpha_t}\mathbf{z}_t \quad \text{for } t = 1, \dots, T \quad (1)$$

where $\alpha_t \in [0, 1]$ governs the noise level, and $\mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ represents Gaussian noise. As t increases, \mathbf{x}_t becomes progressively noisier, ultimately resembling a normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ when $t = T$. The reverse diffusion process is designed to reconstruct the original data from its noisy counterpart:

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)), \quad (2)$$

where μ_θ and Σ_θ are learned parameters defining the mean and covariance.

Timestep Embedding in Diffusion Models. The diffusion procedures are usually splitted to one thousand timesteps during training phase and dozens of timesteps during inference phase. Timestep defines the strength of noise to be added or removed in the diffusion procedures, which is an important input of the diffusion model. Specifically, the scalar timestep t is firstly transformed to timestep embedding through sinusoidal embedding and multilayer perception module:

$$\mathbf{T}_t = MLP(\text{sinusoidal}(t)) \quad \text{for } t = 1, \dots, T. \quad (3)$$

Timestep embedding then modulates the input and output of the Self Attention Layer and Feed Forward Network (FFN) in each Transformer block, as shown in Fig.4. Thus, timestep embedding can significantly affect the magnitude of the model output.

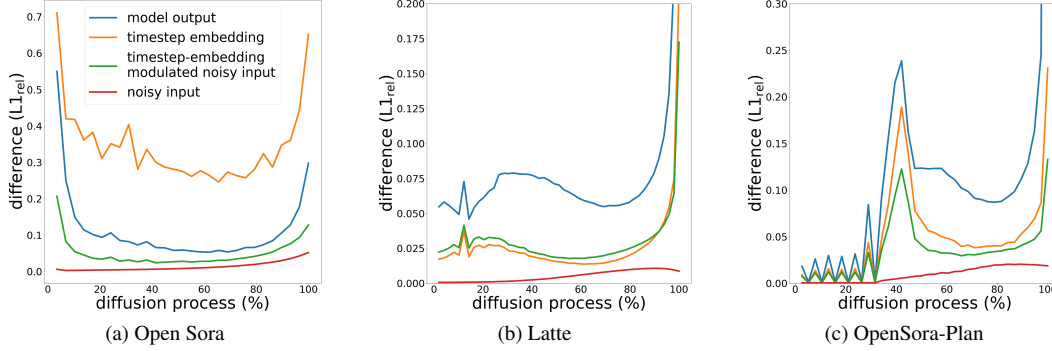


Figure 3. Visualization of input differences and output differences in consecutive timesteps of Open Sora, Latte, and OpenSora-Plan. Timestep embedding and Timestep embedding modulated noisy input have strong correlation with model output.

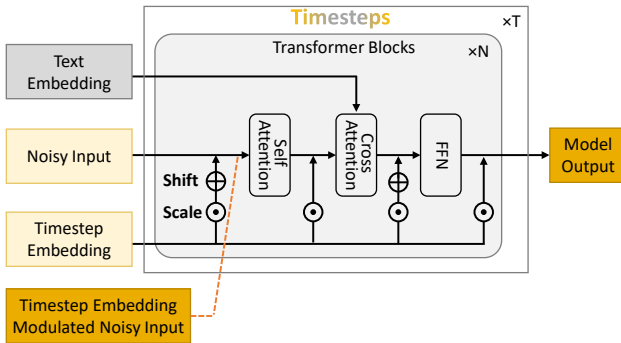


Figure 4. Diffusion module of the visual generation model with transformer. Normalization layer is omitted for simplicity. Timestep embedding modulates the magnitude of block input and output thus has the potential to indicate the variation of output. The output is updated with Eq. 2 at different timesteps.

3.2. Analysis

To investigate the correlation between model output and input, we perform an in-depth analysis of their behaviors during the diffusion process.

Model outputs: Ideally, if we could obtain the model outputs in advance, we could directly measure the difference between outputs at adjacent timesteps and decide whether to cache the outputs based on their difference. Following [56], we use the relative L1 distance as our metric. For instance, the relative L1 distance $L1_{rel}(\mathbf{O}, t)$ for output embedding \mathbf{O}_t at timestep t is calculated as follows:

$$L1_{rel}(\mathbf{O}, t) = \frac{\|\mathbf{O}_t - \mathbf{O}_{t+1}\|_1}{\|\mathbf{O}_{t+1}\|_1} \quad (4)$$

where a large $L1_{rel}(\mathbf{O}, t)$ indicates that \mathbf{O}_t is informative relative to \mathbf{O}_{t+1} and should be cached; otherwise, a small $L1_{rel}(\mathbf{O}, t)$ indicates that \mathbf{O}_{t+1} and \mathbf{O}_t are similar to each other and therefore \mathbf{O}_{t+1} could be reused to replace \mathbf{O}_t . Therefore, Eq. 4 can be used to define a criterion for determining whether the model outputs should be cached.

However, in most cases, the model outputs cannot be obtained in advance, making the above approach infeasible. To address this issue, an intuitive idea is that if we can efficiently estimate the difference of the model outputs, we can leverage it to design a caching strategy. Fortunately, it is well-known that the model inputs and outputs are strongly correlated. Based on this insight, we analyzed the model inputs and conducted detailed experiments to investigate their correlation with the model outputs.

Model inputs: We consider the inputs of diffusion model: text embedding, timestep embedding, and noisy input, as shown in Fig. 4. Since the text embedding remains constant throughout the diffusion process, it can't be used to measure the difference of inputs across timestep. Therefore, text embedding is excluded from analysis. As for the timestep embedding, it changes as timesteps progress but is independent of the noisy input and text embedding, making it difficult to fully reflect the information of the input. The noisy input, on the other hand, is gradually updated during the denoising process and contains information from the text embedding, but it is not sensitive to timesteps. To comprehensively represent the model inputs and ensure their correlation with the outputs, we ultimately utilized the timestep embedding modulated noisy input at the Transformer's input stage as the final input embedding, as illustrated in the Fig. 4.

Experimental analysis: To derive a robust conclusion, we make analysis using the metric defined in Eq. 4 to compute the difference of model inputs and outputs on three distinct video generation models: Open Sora [62], Latte [31], and OpenSora Plan [22]. As illustrated in Fig. 3, the difference of outputs exhibit distinct patterns across various models. In Open Sora, the pattern forms a 'U' shape, whereas in Latte and OpenSora-Plan, it resembles a horizontally flipped 'L'. Additionally, OpenSora-Plan features multiple peaks because its scheduler samples certain timesteps twice. The noisy input across consecutive timesteps changes minimally and shows little correlation with the model output.

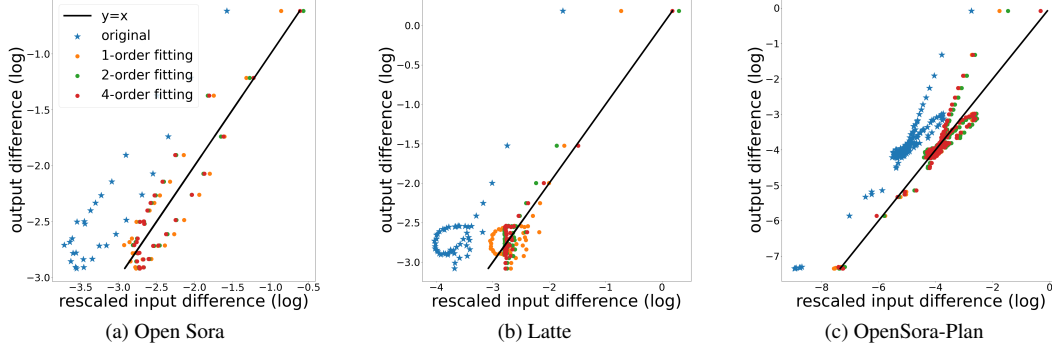


Figure 5. Visualization of correlation of input differences and output differences in consecutive timesteps of Open Sora, Latte, and OpenSora-Plan. The original data points deviate a lot from the linear correlation. Polynomial fitting reduces the gap.

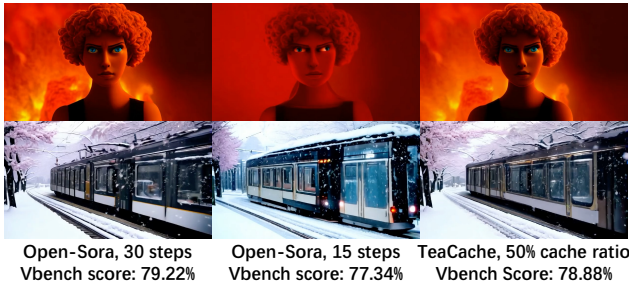


Figure 6. Caching Mechanism v.s. Reducing Timesteps. Reducing inference timesteps suffers from deteriorated visual quality while TeaCache maintains visual quality.

In contrast, both the timestep embedding and the timestep embedding modulated noisy input demonstrate a strong correlation with the model output. Given that the timestep embedding modulated noisy inputs exhibits superior generation capabilities (*e.g.*, in Open Sora) and effectively leverages the dynamics of input, we select it as the indicator to determine whether the model output at the current step is similar to that of the previous timestep.

3.3. TeaCache

As illustrated in Fig. 3, adjacent timesteps conduct redundant computations where model outputs exhibit minimal change. To minimize these redundancies and accelerate inference, we propose the Timestep Embedding Aware Cache (TeaCache). Rather than computing new outputs at each timestep, we reuse cached outputs from previous timesteps. Our caching technique can be applied to nearly all recent diffusion models based on Transformers.

Naive Caching Strategy. To determine whether to reuse the cached model output from a previous timestep, we employ the accumulated relative L1 distance as an indicator.

$$\sum_{t=t_a}^{t_b-1} L1_{\text{rel}}(\mathbf{F}, t) \leq \delta < \sum_{t=t_a}^{t_b} L1_{\text{rel}}(\mathbf{F}, t) \quad (5)$$

where $L1_{\text{rel}}$ is defined in Eq. 4. F can be timestep embedding or timestep embedding modulated noisy inputs and δ is the caching threshold. Specifically, after computing the model output at timestep t_a and caching it, we accumulate the relative L1 distance $\sum_{t=t_a}^{t_b-1} L1_{\text{rel}}(\mathbf{F}, t)$ for subsequent timesteps. If, at timestep $t_b (> t_a)$, $\sum_{t=t_a}^{t_b-1} L1_{\text{rel}}(\mathbf{F}, t)$ is less than the caching threshold δ , we reuse the cached model output; otherwise, we compute the new model output and set the accumulated relative L1 distance to zero. A smaller threshold δ results in more frequent refreshing of cached outputs, while a larger threshold speeds up visual generation but may adversely affect image appearance. The threshold δ should be chosen to enhance inference speed without compromising visual quality.

Rescaled Caching Strategy. Although timestep embedding modulated noisy inputs exhibit a strong correlation with model outputs, the differences in consecutive timesteps are inconsistent. Directly using the difference of timestep embedding-modulated noisy input to estimate model output difference leads to a scaling bias. Such bias may cause suboptimal timestep selection. Considering that these differences are scalars, we apply simple polynomial fitting to rescale them to reduce the bias. The polynomial fitting is then performed between model inputs (timestep embedding modulated noisy inputs) and outputs, which is formulated as

$$y = f(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n, \quad (6)$$

where y represents estimated difference of model output and x signifies the difference of timestep embedding-modulated noisy inputs. This can be efficiently solved using the *poly1d* function from the numpy package. With polynomial fitting, the rescaled difference in timestep embedding-modulated noisy inputs better estimates model output difference, as shown in Fig. 5. The final caching indicator is formulated as

$$\sum_{t=t_a}^{t_b-1} f(L1_{\text{rel}}(\mathbf{F}, t)) \leq \delta < \sum_{t=t_a}^{t_b} f(L1_{\text{rel}}(\mathbf{F}, t)) \quad (7)$$

3.4. Discussion

Caching Mechanism v.s. Reducing Timesteps. Assume that both of the caching mechanism and reducing timesteps strategies reduces half of the timesteps. The differences between them can be concluded in three aspects: (1) Timesteps. Our caching strategy dynamically selects timesteps with large difference for caching and reusing in the following timesteps, whereas the strategy of reducing timesteps is conducted uniformly, lacking awareness of dynamic differences among different timesteps. (2) Model output. In our caching strategy, only the residual signal (*i.e.*, Output minus Input) in the diffusion transformer is cached, therefore the model output at the next timestep is updated. In contrast, reducing timesteps can be considered as keeping the output constant in the next timestep. (3) Parameter α_t . Reducing timesteps results in a coarser-grained α_t , which suffers from deteriorated visual quality. In comparison, TeaCache is able to maintain the visual quality, as illustrated in Fig. 6.

4. Experiment

4.1. Settings

Base Models and Compared Methods. To demonstrate the effectiveness of our method, we apply our acceleration technique to various video, such as Open-Sora 1.2 [62], Open-Sora-Plan [22] and Latte [31]. We compare our base models with recent efficient video synthesis techniques, including PAB [61], T-GATE [60] and Δ -DiT [11], to highlight the advantages of our approach. Notably, Δ -DiT and T-GATE are originally designed as an acceleration method for image synthesis. PAB adapted them for video synthesis to facilitate comparison.

Evaluation Metrics. To assess the performance of video synthesis acceleration methods, we focus on two primary aspects: inference efficiency and visual quality. For evaluating inference efficiency, we use Floating Point Operations (FLOPs) and inference latency as metrics. Following PAB [61], we employ VBench [18], LPIPS [59], PSNR, and SSIM for visual quality evaluation.

Implementation Detail All experiments are carried out on the NVIDIA A800 80GB GPUs with Pytorch. We enable FlashAttention [12] by default for all experiments. To obtain robust polynomial fitting, we sample 70 prompts from T2V-CompBench [49] to generate videos, assessing seven desired attributes of generated videos. 10 prompts are sampled for each attributes. δ in Eq. 7 is 0.1 for TeaCache-slow and 0.2 for TeaCache-fast.

4.2. Main Results

Quantitative Comparison. Tab. 1 presents a quantitative evaluation of efficiency and visual quality using the VBench benchmark [18]. We examine two variants of TeaCache: a

slow variant and a fast variant with greater speedup. Compared to other training-free acceleration methods, TeaCache consistently achieves superior efficiency and better visual quality across different base models, sampling schedulers, video resolutions, and lengths. In evaluating the Latte [31] baseline, the TeaCache-slow model demonstrates superior performance across all visual quality metrics, achieving a 1.86 \times speedup compared to PAB [61], which provides a 1.34 \times speedup. TeaCache-fast achieves the highest acceleration at 3.28 \times , albeit with a slight reduction in visual quality. With the OpenSora [62] baseline, we obtain the optimal speedup of 2.25 \times as compared to the previous 1.40 \times , and the highest overall quality with a speedup of 1.55 \times . Additionally, using Open-Sora-Plan [22], TeaCache achieves the highest speedup of 6.83 \times , surpassing the previously best 1.49 \times offered by PAB, while also delivering the highest quality at a speedup of 4.41 \times .

Visualization. Fig. 7 compares the videos generated by TeaCache against those by the original model and PAB. The results demonstrate that TeaCache outperforms PAB in visual quality with lower latency.

4.3. Ablation Studies

Scaling to multiple GPUs. Aligned with previous research employing Dynamic Sequence Parallelism (DSP) [61] for supporting high-resolution long-video generation across multiple GPUs, we assess the performance of TeaCache in these scenarios. The results of this study are presented in Tab. 4. We utilize Open-Sora [62] and Open-Sora-Plan [22] as baselines and compare them against the prior method PAB [61] regarding latency measurements on A800 GPUs. As the number of GPUs increases, TeaCache consistently improves inference speed across various base models and outperforms PAB.

Performance at different Length and Resolution. To assess the effectiveness of our method in accelerating sampling for videos with varying sizes, we perform tests across different video lengths and resolutions. The results, presented in Fig. 8, demonstrate that our method sustains consistent acceleration performance, even with increases in video resolution and frame count. This consistency highlights the method’s potential to accelerate sampling processes for longer and higher-resolution videos, meeting practical demands.

Quality-Efficiency trade-off. In Fig. 1, we compare the quality-latency trade-off of TeaCache with PAB [61]. The thresholds δ in Eq. 7 are 0.1, 0.15, 0.2 and 0.25. Our analysis reveals that TeaCache achieves significantly higher reduction rates, indicated by lower absolute latency, compared to PAB. Additionally, across a wide range of latency configurations, TeaCache consistently outperforms PAB on all quality metrics. This is particularly evident in the reference-free metric VBench score [18], which aligns

Table 1. Quantitative evaluation of inference efficiency and visual quality in video generation models. TeaCache consistently achieves superior efficiency and better visual quality across different base models, sampling schedulers, video resolutions, and lengths.

Method	Efficiency			Visual Quality			
	FLOPs (P) ↓	Speedup ↑	Latency (s) ↓	VBench ↑	LPIPS ↓	SSIM ↑	PSNR ↑
Latte (16 frames, 512×512)							
Latte ($T = 50$)	3.36	1×	26.90	77.40%	-	-	-
Δ -DiT [11]	3.36	1.02×	-	52.00%	0.8513	0.1078	8.65
T-GATE [60]	2.99	1.13×	-	75.42%	0.2612	0.6927	19.55
PAB-slow [61]	2.70	1.21×	22.16	76.32%	0.2669	0.7014	19.71
PAB-fast [61]	2.52	1.34×	19.98	73.13%	0.3903	0.6421	17.16
TeaCache-slow	1.86	1.86 ×	14.46	77.40%	0.1901	0.7786	22.09
TeaCache-fast	1.12	3.28 ×	8.20	76.69%	0.3133	0.6678	18.62
Open-Sora 1.2 (51 frames, 480P)							
Open-Sora 1.2 ($T = 30$)	3.15	1×	44.56	79.22%	-	-	-
Δ -DiT [11]	3.09	1.03×	-	78.21%	0.5692	0.4811	11.91
T-GATE [60]	2.75	1.19×	-	77.61%	0.3495	0.6760	15.50
PAB-slow [61]	2.55	1.33×	33.40	77.64%	0.1471	0.8405	24.50
PAB-fast [61]	2.50	1.40×	31.85	76.95%	0.1743	0.8220	23.58
TeaCache-slow	2.40	1.55×	28.78	79.28%	0.1316	0.8415	23.62
TeaCache-fast	1.64	2.25×	19.84	78.48%	0.2511	0.7477	19.10
OpenSora-Plan (65 frames, 512×512)							
OpenSora-Plan ($T = 150$)	11.75	1×	99.65	80.39%	-	-	-
Δ -DiT [11]	11.74	1.01×	-	77.55%	0.5388	0.3736	13.85
T-GATE [60]	2.75	1.18×	-	80.15%	0.3066	0.6219	18.32
PAB-slow [61]	8.69	1.36×	73.41	80.30%	0.3059	0.6550	18.80
PAB-fast [61]	8.35	1.56×	65.38	71.81%	0.5499	0.4717	15.47
TeaCache-slow	3.13	4.41×	22.62	80.32%	0.2145	0.7414	21.02
TeaCache-fast	2.06	6.83×	14.60	79.72%	0.3155	0.6589	18.95

more closely with human preferences. Although there is a decline in reference-based scores such as PSNR and SSIM at extreme reduction rates, qualitative results suggest that the outputs remain satisfactory, despite not perfectly matching the reference.

Choice of Indicator. When determining the caching schedule, we evaluate various indicators to estimate the differences in model outputs across consecutive timesteps, including timestep embedding and timestep embedding-modulated noisy input. As illustrated in Fig. 3, the timestep embedding-modulated noisy input demonstrates a stronger correlation with model output compared to the timestep embedding, particularly in the OpenSora. Moreover, the selection of timesteps by the timestep embedding-modulated noisy input adapts dynamically to different prompts, whereas the timestep embedding selects the same timesteps for all prompts. This observation is validated by the results presented in Tab. 2, where the timestep embedding-modulated noisy input consistently surpasses the timestep embedding across various models.

Effect of Rescaling. Tab.3 illustrates the impact of

Table 2. Ablation study of caching indicator. ‘Timestep’: timestep embedding. ‘Input’: timestep embedding-modulated noisy input.

Indicator	VBench ↑	LPIPS ↓	SSIM ↑	PSNR ↑
OpenSora	79.22%	-	-	-
Timestep	77.01%	0.3425	0.6934	15.86
Input	78.21%	0.2549	0.7457	19.05
Latte	77.40%	-	-	-
Timestep	77.05%	0.2653	0.7073	19.76
Input	77.17%	0.2558	0.7164	20.00

rescaling. A first-order polynomial fitting outperforms the original data by 0.24% under Vbench score metric, as well as in LPIPS, SSIM, and PSNR metrics. Performance gains tend to saturate with a fourth-order polynomial fitting.

5. Conclusion

In this study, we introduce **TeaCache**, a novel, training-free approach designed to significantly accelerate video synthesis inference while maintaining high-quality output. We analyze the correlation between model input and output, observing that similarity of timestep embedding modulated noisy input in consecutive timesteps shows strong correla-

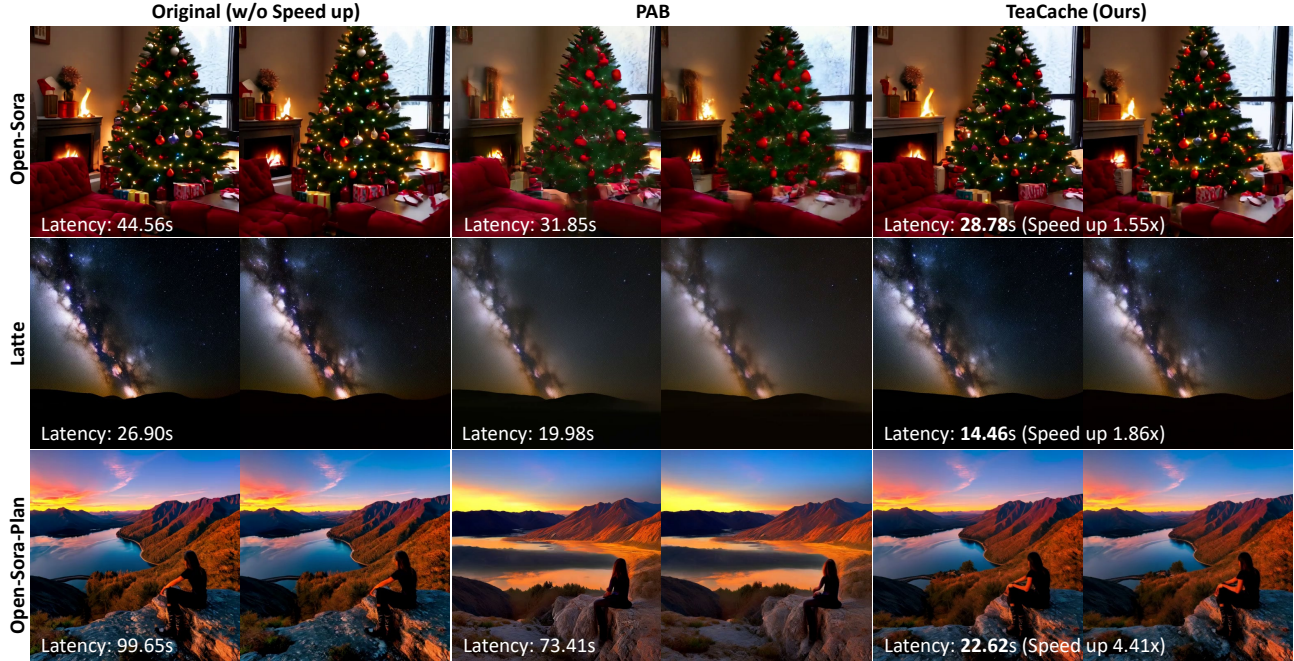


Figure 7. Comparison of visual quality and efficiency (denoted by latency) with the competing method. TeaCache outperforms PAB [61] in both visual quality and efficiency. Latency is evaluated on a single A800 GPU. Video generation specifications: Open-Sora [62] (51 frames, 480p), Latte [31] (16 frames, 512×512), Open-Sora-Plan [22] (65 frames, 512×512). Best-viewed with zoom-in.

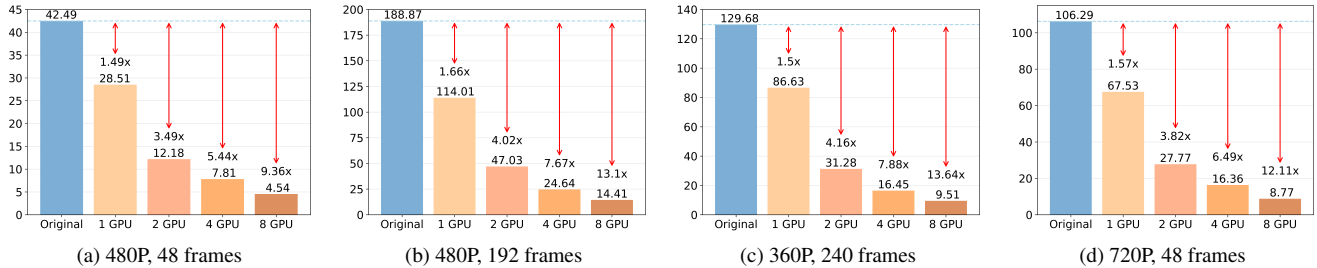


Figure 8. Inference efficiency of TeaCache at different video lengths and resolutions.

Table 3. Ablation study of polynomial fitting. Rescaling with polynomial fitting outperforms original data. Higher-order fitting obtains better performance and saturates in 4-order fitting.

Order	VBench \uparrow	LPIPS \downarrow	SSIM \uparrow	PSNR \uparrow
OpenSora	79.22%	-	-	-
Original	78.21%	0.2549	0.7457	19.05
1-order	78.45%	0.2517	0.7478	19.10
2-order	78.48%	0.2513	0.7477	19.09
4-order	78.48%	0.2511	0.7477	19.10

tion with similarity of model output. We propose to utilize similarity of timestep embedding modulated noisy input as an indicator of output similarity, allowing for dynamic caching of model outputs. Further, we propose a rescaling strategy to refine the estimation of model output similarity, optimizing the selection of timestep. Extensive experi-

Table 4. Inference efficiency and visual quality when scaling to multiple GPUs with Dynamic Sequence Parallelism (DSP).

Method	1 × A800	2 × A800	4 × A800	8 × A800
Open-Sora (192 frames, 480P)				
Baseline	188.87(1×)	72.86(2.59×)	39.26(4.81×)	22.18(8.52×)
PAB	142.23(1.33×)	53.74(3.51×)	29.19(6.47×)	16.88(11.19×)
TeaCache	114.01(1.66×)	47.03(4.02×)	24.64(7.67×)	14.41(13.10×)
Open-Sora-Plan (221 frames, 512×512)				
Baseline	324.41(1×)	166.94(1.94×)	88.18(3.68×)	47.79(6.79×)
PAB	207.70(1.56×)	110.06(2.95×)	58.07(5.59×)	31.92(10.16×)
TeaCache	48.22(6.73×)	26.99(12.02×)	15.91(20.39×)	10.13(32.02×)

ments demonstrate TeaCache’s robust performance in terms of both efficiency and visual quality across diverse video generation models and image generation models, sampling schedules, video lengths, and resolutions, underscoring its potential for real-world applications.

Acknowledgment. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62472402, the Fundamental Research Funds for the Central Universities, National Natural Science Foundation of China (NSFC) under Grant 62225208 and CAS Project for CAS Project for Young Scientists in Basic Research under Grant No.YSBR-117 and Alibaba Research Intern Program.

References

- [1] Mochi 1, 2024. <https://www.genmo.ai>. 1
- [2] Vchitect. vchitect-2.0: Parallel transformer for scaling up video diffusion models, 2024. <https://github.com/Vchitect/Vchitect-2.0>. 1
- [3] David H Albonesi. Selective cache ways: On-demand cache resource allocation. In *MICRO-32. Proceedings of the 32nd Annual ACM/IEEE International Symposium on Microarchitecture*, pages 248–259. IEEE, 1999. 2
- [4] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 1
- [5] Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4599–4603, 2023. 3
- [6] Haoxin Chen, Menghan Xia, Yingqing He, Yong Zhang, Xiaodong Cun, Shaoshu Yang, Jinbo Xing, Yaofang Liu, Qifeng Chen, Xintao Wang, et al. Videocrafter1: Open diffusion models for high-quality video generation. *arXiv preprint arXiv:2310.19512*, 2023. 3
- [7] Haoxin Chen, Yong Zhang, Xiaodong Cun, Menghan Xia, Xintao Wang, Chao Weng, and Ying Shan. Videocrafter2: Overcoming data limitations for high-quality video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7310–7320, 2024. 3
- [8] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, et al. Pixart-alpha: Fast training of diffusion transformer for photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426*, 2023. 3
- [9] Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-sigma: Weak-to-strong training of diffusion transformer for 4k text-to-image generation. *arXiv preprint arXiv:2403.04692*, 2024. 1, 3
- [10] Lei Chen, Yuan Meng, Chen Tang, Xinzhu Ma, Jingyan Jiang, Xin Wang, Zhi Wang, and Wenwu Zhu. Q-dit: Accurate post-training quantization for diffusion transformers. *arXiv preprint arXiv:2406.17343*, 2024. 2
- [11] Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. Delta dit: A training-free acceleration method tailored for diffusion transformers. *arXiv preprint arXiv:2406.01125*, 2024. 2, 3, 6, 7
- [12] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022. 6
- [13] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 1
- [14] James R Goodman. Using cache memory to reduce processor-memory traffic. In *Proceedings of the 10th annual international symposium on Computer architecture*, pages 124–131, 1983. 2
- [15] Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Ptdq: Accurate post-training quantization for diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024. 3
- [16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1, 3
- [17] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022. 3
- [18] Ziqi Huang, Yanan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21807–21818, 2024. 6
- [19] Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*, 2021. 3
- [20] Kumara Kahatapitiya, Haozhe Liu, Sen He, Ding Liu, Menglin Jia, Chenyang Zhang, Michael S Ryoo, and Tian Xie. Adaptive caching for faster video generation with diffusion transformers. *arXiv preprint arXiv:2411.02397*, 2024. 3
- [21] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022. 3
- [22] PKU-Yuan Lab and Tuzhan AI etc. Open-sora: Democratizing efficient video production for all, 2024. <https://doi.org/10.5281/zenodo.10948109>. 1, 3, 4, 6, 8
- [23] Muyang Li, Tianle Cai, Jiaxin Cao, Qinsheng Zhang, Han Cai, Junjie Bai, Yangqing Jia, Kai Li, and Song Han. Distribution: Distributed parallel inference for high-resolution diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7183–7193, 2024. 3
- [24] Senmao Li, Taihang Hu, Fahad Shahbaz Khan, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, and Jian Yang. Faster diffusion: Rethinking the role of unet encoder in diffusion models. *arXiv e-prints*, pages arXiv–2312, 2023. 3
- [25] Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Snapfusion: Text-to-image diffusion model on mobile devices

- within two seconds. *Advances in Neural Information Processing Systems*, 36, 2024. 1
- [26] Yanjing Li, Sheng Xu, Xianbin Cao, Xiao Sun, and Baochang Zhang. Q-dm: An efficient low-bit quantized diffusion model. *Advances in Neural Information Processing Systems*, 36, 2024. 3
- [27] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022. 3
- [28] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022. 3
- [29] Zhengyao Lv, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K Wong. Fastercache: Training-free video diffusion model acceleration with high quality. *arXiv preprint arXiv:2410.19355*, 2024. 3
- [30] Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. *arXiv preprint arXiv:2406.01733*, 2024. 2
- [31] Xin Ma, Yaohui Wang, Gengyun Jia, Xinyuan Chen, Ziwei Liu, Yuan-Fang Li, Cunjian Chen, and Yu Qiao. Latte: Latent diffusion transformer for video generation. *arXiv preprint arXiv:2401.03048*, 2024. 1, 3, 4, 6, 8
- [32] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14297–14306, 2023. 2
- [33] OpenAI. Sora, 2024. <https://openai.com/index/sora/>. 3
- [34] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023. 1, 3
- [35] Aditya Ramesh, Prfulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1 (2):3, 2022. 1, 3
- [36] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.
- [37] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022. 1, 3
- [38] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 3
- [39] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023. 2
- [40] Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. Fora: Fast-forward caching in diffusion transformer acceleration. *arXiv preprint arXiv:2407.01425*, 2024. 2, 3
- [41] Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1972–1981, 2023. 3
- [42] Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024. 1
- [43] Alan Jay Smith. Cache memories. *ACM Computing Surveys (CSUR)*, 14(3):473–530, 1982. 2
- [44] Junhyuk So, Jungwon Lee, and Eunhyeok Park. Frdiff: Feature reuse for universal training-free acceleration of diffusion models. *arXiv preprint arXiv:2312.03517*, 2023. 3
- [45] Junhyuk So, Jungwon Lee, Daehyun Ahn, Hyungjun Kim, and Eunhyeok Park. Temporal dynamic quantization for diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024. 3
- [46] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 1, 3
- [47] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 3
- [48] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 1, 3
- [49] Kaiyue Sun, Kaiyi Huang, Xian Liu, Yue Wu, Zihan Xu, Zhenguo Li, and Xihui Liu. T2v-compbench: A comprehensive benchmark for compositional text-to-video generation. *arXiv preprint arXiv:2407.14505*, 2024. 6
- [50] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 3
- [51] Hongjie Wang, Difan Liu, Yan Kang, Yijun Li, Zhe Lin, Niraj K Jha, and Yuchen Liu. Attention-driven training-free efficiency enhancement of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16080–16089, 2024. 3
- [52] Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. Modelscope text-to-video technical report. *arXiv preprint arXiv:2308.06571*, 2023. 3
- [53] Xiang Wang, Shiwei Zhang, Han Zhang, Yu Liu, Yingya Zhang, Changxin Gao, and Nong Sang. Videolcm: Video latent consistency model. *arXiv preprint arXiv:2312.09109*, 2023. 2, 3
- [54] Yujie Wei, Shiwei Zhang, Zhiwu Qing, Hangjie Yuan, Zhiheng Liu, Yu Liu, Yingya Zhang, Jingren Zhou, and Hongming Shan. Dreamvideo: Composing your dream videos with customized subject and motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6537–6549, 2024. 3

- [55] Yujie Wei, Shiwei Zhang, Hangjie Yuan, Xiang Wang, Haonan Qiu, Rui Zhao, Yutong Feng, Feng Liu, Zhizhong Huang, Jiaxin Ye, et al. Dreamvideo-2: Zero-shot subject-driven video customization with precise motion control. *arXiv preprint arXiv:2410.13830*, 2024. 3
- [56] Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, Xiaoliang Dai, Ji Hou, Zijian He, Artsiom Sanakoyeu, Peizhao Zhang, Sam Tsai, Jonas Kohler, et al. Cache me if you can: Accelerating diffusion models through block caching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6211–6220, 2024. 3, 4
- [57] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu. Deepcache: Principled cache for mobile deep vision. In *Proceedings of the 24th annual international conference on mobile computing and networking*, pages 129–144, 2018. 2, 3
- [58] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024. 1, 3
- [59] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6
- [60] Wentian Zhang, Haozhe Liu, Jinheng Xie, Francesco Facio, Mike Zheng Shou, and Jürgen Schmidhuber. Cross-attention makes inference cumbersome in text-to-image diffusion models. *arXiv preprint arXiv:2404.02747*, 2024. 3, 6, 7
- [61] Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. *arXiv preprint arXiv:2408.12588*, 2024. 1, 2, 3, 6, 7, 8
- [62] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all, 2024. <https://github.com/hpcaitech/Open-Sora>. 1, 3, 4, 6, 8