Model Reveals What to Cache: Profiling-Based Feature Reuse for Video Diffusion Models

Xuran Ma[1,2*] Yexin Liu[1,2*] Yaofu Liu[2] Xianfeng Wu[1,2] Mingzhe Zheng[1,2] Zihao Wang[1,2]
Ser-Nam Lim[2,3†] Harry Yang[1,2†]

[1]Hong Kong University of Science and Technology [2]Everlyn AI [3]University of Central Florida
[*]Equal Contribution [†]Corresponding Author

xmacb@connect.ust.hk, yliu292@connect.ust.hk, sernam@gmail.com, harryyang.hk@gmail.com

GitHub: https://github.com/GeekGuru123/ProfilingDiT/tree/main

Figure 1. Comparison of visual result quality across different methods. The video has a resolution of 780p and consists of 129 frames, with one representative frame extracted from each video for visualization. Our method consistently outperforms TeaCache [15] in both visual quality and efficiency. Latency is evaluated using a single H200 GPU. All results are generated with seed 42.

Abstract

Recent advances in diffusion models have demonstrated remarkable capabilities in video generation. However, the computational intensity remains a significant challenge for practical applications. While feature caching has been proposed to reduce the computational burden of diffusion models, existing methods typically overlook the heterogeneous significance of individual blocks, resulting in suboptimal reuse and degraded output quality. To this end, we address this gap by introducing ***ProfilingDiT****, a novel adaptive caching strategy that explicitly disentangles foreground and background-focused blocks. Through a systematic analysis of attention distributions in diffusion models, we reveal a key observation: 1) Most layers exhibit a consistent preference for either foreground or background regions. 2) Predicted noise shows low inter-step similarity initially, which stabilizes as denoising progresses. This finding inspires us to formulate a selective caching strategy that preserves full computation for dynamic foreground elements while efficiently caching static background features. Our approach substantially reduces computational overhead while preserving visual fidelity. Extensive experiments demonstrate that our framework achieves significant acceleration (e.g., 2.01× speedup for Wan2.1) while maintaining visual fidelity across comprehensive quality metrics, establishing a viable method for efficient video generation.*

1. Introduction

Video generation has emerged as a pivotal technology in computer vision, driving advancements across diverse domains from multimedia applications and entertainment to interactive systems [2, 6, 15, 21, 25, 27, 36]. This surge in demand has driven rapid progress in generative modeling [3, 29, 33], particularly diffusion-based approaches. Among these methods, the Diffusion Transformer (DiT) architecture [26] has gained prominence for its ability to synthesize high-fidelity and temporally consistent videos, supporting downstream tasks including augmented reality [8], automated content creation [34, 37], and AI-driven storytelling [10, 39, 41]. However, despite their remarkable success in visual quality and diversity, DiT models face a critical challenge: their inherently sequential denoising process necessitates iterative computations, resulting in substantial computational overhead and substantial memory demands, particularly for high-resolution video generation and large-scale models.

To address this computational bottleneck, researchers have explored various acceleration strategies, including model distillation [16, 18], quantization [4], and caching methods. Among these approaches, caching methods have shown particular promise due to their training-free nature and compatibility with pretrained models. Recent advances in this direction include feature reuse mechanisms [19], attention-based caching [20, 40], and adaptive routing strategies [14, 28]. However, these methods predominantly rely on input-output similarity metrics for cache decisions, overlooking the heterogeneous importance of different blocks. As a result, caching blocks with high similarity may still lead to significant information loss if those blocks play a critical role in semantic modeling. Through systematic analysis of the DiT architecture, we reveal two critical insights: (1) Most layers exhibit a consistent preference for either foreground or background regions. (2) In the initial denoising steps, the predicted noise exhibits low inter-step similarity, gradually stabilizes in the later stages.

Motivated by these findings, we introduce **ProfilingDiT**, a novel adaptive caching framework that explicitly disentangles foreground and background-focused computations. Our approach leverages the observed semantic separation in transformer blocks to implement a dual-granularity caching strategy. At the block level, we selectively cache features from layers that predominantly attend to background regions, while preserving full computation for those focused on foreground details. At the step level, we analyze the L1 distance between consecutive denoising steps as a proxy for prediction stability. In the early steps, the distance is high, reflecting rapid semantic transitions. Thus, caching is disabled to retain full update precision. As diffusion progresses, the distance stabilizes, allowing efficient caching at larger intervals. In the final steps, the distance increases again, indicating the need for finer-grained updates and a reduced caching rate.

Our comprehensive experiments demonstrate the effectiveness of ProfilingDiT. For instance, on the Wan2.1 [31] model, our approach achieves a 2.01× speed-up in inference while maintaining a competitive LPIPS of 0.1256. In addition to perceptual quality, we also observe substantial gains in other metrics, with PSNR improved to 22.02 and SSIM reaching 0.7899. These results confirm that ProfilingDiT enables low-latency generation without compromising visual fidelity.

The primary contributions of our work include:

- **Attention Analysis of DiT in diffusion:** We investigate the attention distributions and noise dynamics in DiT during the denoise process, revealing consistent foreground-background preferences across blocks and temporally evolving noise similarity across diffusion steps.
- **Adaptive Caching Framework:** We introduce a novel dual-granularity caching strategy that leverages semantic separation for optimal computational resource allocation, significantly advancing the state-of-the-art in efficient video generation.
- **Empirical Validation:** Through extensive experimentation, we demonstrate that ProfilingDiT achieves substantial computational efficiency (up to 2.01× acceleration) while maintaining high-quality video generation across diverse evaluation metrics.

2. Related Works

Video Diffusion Model. Diffusion models [3, 11, 22, 30] have emerged as foundational frameworks for generative tasks, showcasing remarkable capabilities in producing diverse and high-quality outputs. Initially employing U-Net architectures, these models have been highly successful in both image and video generation tasks. However, the inherent scalability limitations of U-Net-based diffusion models [9, 32] restrict their effectiveness for complex, large-scale generation tasks. To overcome this constraint,DiT was introduced, effectively leveraging the scalability and representational power of transformer architectures. Notably, models like Sora have demonstrated significant advancements by employing transformer-based diffusion frameworks to capture complex dynamics of real-world phenomena. Recent research has further evolved video diffusion models from initial 2D architectures with additional temporal modeling [12, 42] to fully 3D diffusion frameworks [15, 17, 24, 31, 35]. This architectural shift enables better handling of spatial-temporal dependencies, significantly improving video quality, motion coherence, and generation fidelity over extended sequences.

Caching Methods for Video Diffusion Transformers. Accelerating video diffusion model inference remains an

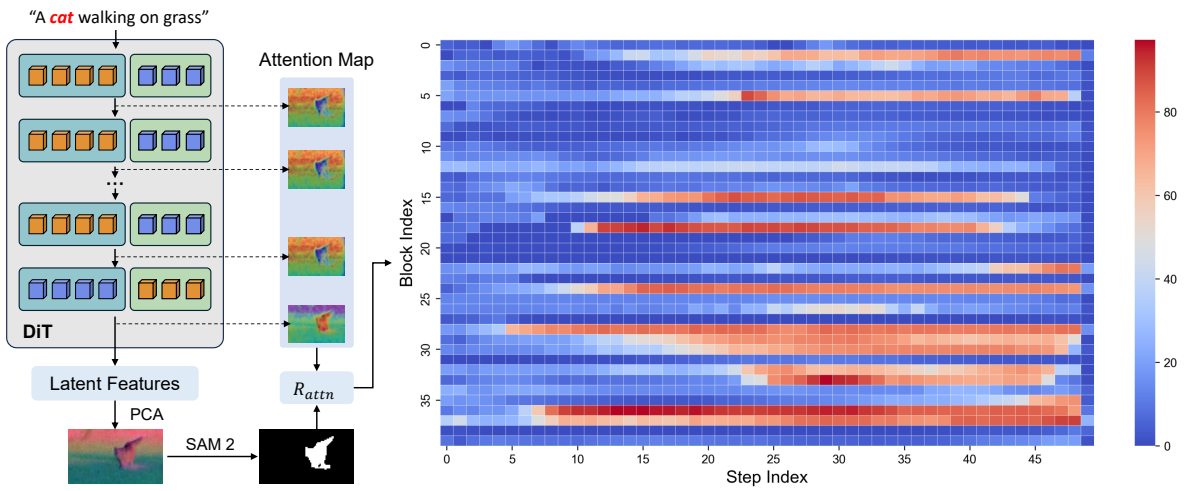


Figure 2. **Visualization of attention focus across blocks and diffusion steps in DiT.** *Left*: We compute the foreground attention ratio R_{attn} for each block using attention maps and semantic masks derived from SAM2. *Right*: Heatmap of R_{attn} across 40 blocks and 50 denoising steps in HunyuanVideo [15]. Warmer colors indicate higher foreground focus. Details are in Sec. 3.2.

active area of research, with strategies such as distillation, quantization, and caching being explored extensively. Distillation approaches [16, 18] typically aim to compress larger models into smaller, efficient versions, whereas quantization methods [4] reduce computational overhead by lowering numerical precision. In contrast, caching methods primarily focus on feature reuse during inference to minimize redundant computations. Existing caching approaches for Diffusion Transformers have made notable strides in enhancing inference speed. For instance, TeaCache [19] reuses noise features based on a routing decision mechanism. Methods like PAB [40] and TGate [20] specifically reuse attention outputs by identifying critical attention components. Delta [5] focuses on reusing block outputs, while AdaCache and Fora [14, 28] introduce adaptive routers to optimize reuse across attention and MLP layers. Other advancements include FasterCache [23], which proposes a classifier-free guidance (CFG) caching strategy, and ToCa [43], introducing token-level reuse and compression. However, the transition to fully 3D transformer architectures significantly increases memory usage, making attention-based caching less viable and consequently positioning block-level and noise reuse as preferable methods for managing computational resources in video diffusion models.

3. Methodology

3.1. Preliminaries

Denoising Diffusion Models. Diffusion models simulate visual generation through a sequence of iterative denoising steps. Starting from random noise, these models progressively refine the noise until it closely approximate samples from the desired distribution. During the forward diffusion process, Gaussian noise is incrementally added over T steps to a data point x_0 sampled from the real distribution $q(x)$:

$$x_t = \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t} z_t, \quad t = 1, \ldots, T, \tag{1}$$

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

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)), \tag{2}$$

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

Attention Mechanism. The attention mechanism in DiT operates by computing Query ($\mathbf{Q}$), Key ($\mathbf{K}$), and Value ($\mathbf{V}$) matrices from the input latent features $\mathbf{X}$. These matrices are obtained through learned linear projections:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}^K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}^V \tag{3}$$

where $\mathbf{W}^Q$, $\mathbf{W}^K$, and $\mathbf{W}^V$ are the projection matrices for queries, keys, and values, respectively. The attention scores are computed as follows:

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) \tag{4}$$

where $\mathbf{A}$ is the attention matrix, with A_{ij} representing the attention score of token j attending to token i.

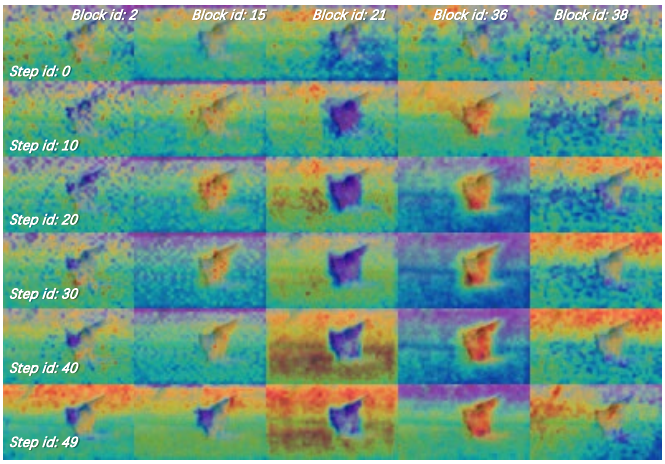


Figure 3. Heatmap of attention scores. Attention scores are displayed at the block level (rows) and step level (columns), revealing consistent patterns across steps. Red denotes regions with the highest attention, followed by green and blue, while purple indicates areas with the lowest attention.

3.2. Attention Pattern Analysis

Given an input prompt, such as *"a cat walking on grass"*, we analyze the attention behavior within DiT blocks. Each block contains a self-attention layer, from which token embeddings $\mathbf{X} \in \mathbb{R}^{B\times N\times C}$ are extracted across all denoising steps. The token dimension N is divided into T segments, each corresponding to one video frame. Attention scores are computed for each frame by summing across rows in the attention matrix.

Block-Level Analysis. To determine whether each block predominantly attends to foreground or background regions, we first extract the noise prediction at each step (shape: $B \times C \times T \times H \times W$). To segment the latent space into foreground and background, we apply PCA [1] to reduce the channel dimension to 3, forming RGB-like visualizations as shown in Fig. 2. We then use SAM2 to generate binary masks of the foreground.

Tokens with attention scores exceeding a predefined threshold are selected. Given a transformer block's attention matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, where N denotes the number of spatial-temporal tokens, we compute the aggregated attention score for each token i as:

$$\bar{a}_i = \frac{1}{N}\sum_{j=1}^{N} A_{ij} \tag{5}$$

The proportion of these high-attention tokens that lie within the foreground mask is computed. Although a single frame is shown for illustration, the calculation is performed over all frames in a block and averaged to yield a score R_{attn}, reflecting the concentration of high-attention tokens within foreground regions.

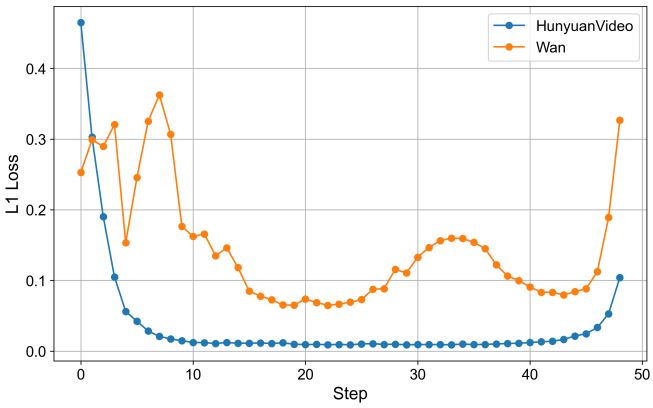


Figure 4. L1 distance of predicted noise on different steps.

$$R_{attn} = \frac{N_{\text{high}\cap\text{fg}}}{N_{\text{fg}}} \tag{6}$$

where $N_{\text{high}\cap\text{fg}}$ denotes the number of tokens that overlap between high-attention($\bar{a}_i >$ threshold) tokens and the foreground mask, and N_{fg} is the total number of foreground tokens. Then we can get a block list from: We define the foreground ($\mathcal{F}$) and background ($\mathcal{B}$) blocks list based on a predefined threshold τ:

$$\mathcal{F} = [i \mid R_{attn} \geq \tau], \quad \mathcal{B} = [i \mid R_{attn} < \tau] \tag{7}$$

Fig. 2 and Fig. 3 visualize the score R_{attn} of 40 single blocks of MMDiT within the HunyuanVideo [15] on all 50 steps. We observe that R_{attn} for each block remains consistent across steps, supporting the hypothesis that blocks exhibit an intrinsic focus on either foreground or background. This attention preference remains stable throughout the denoising process. The attention distribution provides insights into how each block prioritizes different regions of the input. Specifically, some blocks consistently focus on foreground content, while others emphasize the background. Notably, early steps are more background-oriented, with a gradual shift toward foreground attention after step 6.

Step-Level Analysis. We extract noise predictions at different steps from HunyuanVideo [15] and Wan2.1 [31] and compute step-level similarity. Fig. 4 shows the L1 distance between consecutive noise predictions during denoising steps. In the early steps, the L1 distance is high, indicating rapid changes in the predictions. Accordingly, no caching is applied to allow precise updates. In later steps, the L1 distance stabilizes, enabling efficient caching with larger intervals. In the final steps, the L1 distance increases, implying less caching rate.

3.3. ProfilingDiT

We introduce ProfilingDiT, a semantically aware adaptive caching framework that optimizes computational efficiency

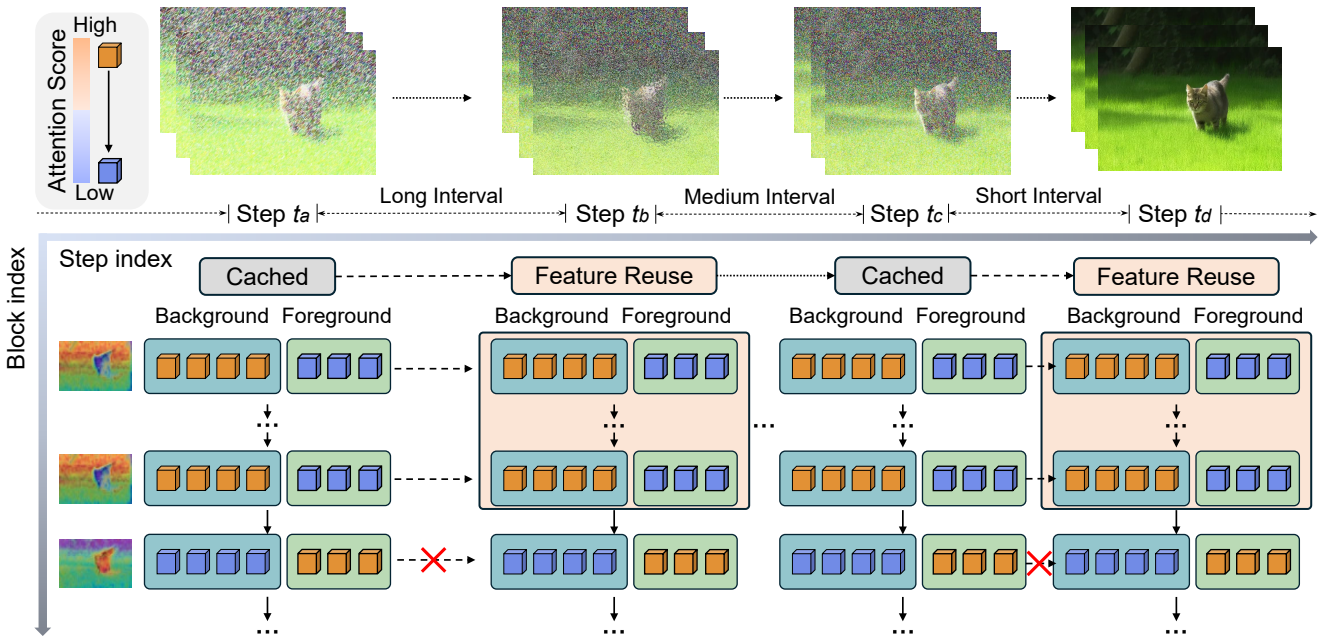


Figure 5. Pipeline of ProfilingDiT: Tokens highlighted in yellow and blue indicate higher and lower attention scores, respectively. The alignment between attention regions and object regions determines whether the corresponding block outputs should be cached.

by explicitly distinguishing between foreground and background processing in diffusion transformers. As shown in Fig. 5, our approach implements a dual-granularity caching strategy, operating at both block and temporal levels to maximize computational resource utilization while preserving generation quality.

Block-Level Caching. Motivated by Delta-DiT [5], we cache block-level deviations between the output and input hidden states. Specifically, for a block i, we compute:

$$\Delta_{\text{block}} = \begin{cases} h_{\text{out}}^{(i)} - h_{\text{in}}^{(i)}, & \text{if } i \in \mathcal{B}, \\ 0, & \text{if } i \in \mathcal{F}. \end{cases} \tag{8}$$

However, based on our previous observations, the block focus on the background list contains both consecutive numbers and individual numbers. The Delta-DiT only caches a single group of consecutive blocks, but we need to process all the blocks in the list. Therefore, we have designed the following *delta list* scheme, as shown in Fig. 6.

$$\Delta_i, \Delta_k, \cdots = \begin{cases} h_{\text{out}}^{(i)} - h_{\text{in}}^{(i)}, & \text{if } i \in \mathcal{B}, \\ h_{\text{out}}^{(j)} - h_{\text{in}}^{(k)}, & \text{if } i \in \mathcal{B},\ j - k > 1, \\ 0, & \text{if } i \in \mathcal{F}. \end{cases} \tag{9}$$

where j and k denote the start and end indices of a continuous interval. and i is an individual number.

To propagate cached features across multiple steps, we maintain an accumulated cache state $\mathcal{C}_t^{(i)}$ at each step t,

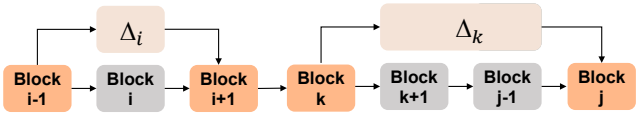


Figure 6. Details of delta list scheme in Sec.3.3.

which updates recursively as follows:

$$\mathcal{C}_{t+1}^{(i)} = \mathcal{C}_t^{(i)} + \Delta_{\text{t}}^{(i)} \tag{10}$$

where $\mathcal{C}_0^{(i)} = 0$ for all blocks at the initial step. This can selectively accumulate deviations for background-focused blocks and optimize computation for continuous block sequences, achieving a balance between computational efficiency and generation quality.

Step-Level Caching. Motivated by the trend observed in Fig. 4, where the L_1 loss is larger in earlier steps and stabilizes over time. We also introduce a temporal caching strategy that distinguishes between cache computation and cache utilization phases. For a given diffusion step s, we define a caching interval T_s that modulates the frequency of cache updates:

$$T_s = f(s), \tag{11}$$

where $f(s)$ is a monotonically decreasing function, ensuring that caching occurs less frequently at the beginning and more frequently in later steps.

For steps s beyond an initial warm-up phase (i.e., $s > s_0$), we maintain two distinct step sets: cache computation

Method	Visual Quality Evaluation					Efficiency Evaluation	
	VBench ↑	LPIPS ↓	PSNR ↑	SSIM ↑	FID ↓	Latency (ms) ↓	Speedup ↑
HunyuanVideo (720P,129frames) [15]	**0.7703**	–	–	–	–	1745	–
TeaCache (slow) [19]	0.7700	0.1720	21.91	0.7456	77.67	1052	1.66×
TeaCache (fast) [19]	0.7677	0.1830	21.60	0.7323	83.85	**753**	**2.31**×
Ours	0.7642	**0.1203**	**26.44**	**0.8445**	**41.10**	932	1.87×

Table 1. Quantitative comparison with prior methods under HunyuanVideo baselines. "Ours" consistently outperforms others in visual quality while maintaining strong efficiency. ↑: higher is better; ↓: lower is better.

Method	Visual Quality Evaluation					Efficiency Evaluation	
	VBench ↑	LPIPS ↓	PSNR ↑	SSIM ↑	FID ↓	Latency (ms) ↓	Speedup ↑
Wan2.1 (480P,81frames) [31]	0.7582	–	–	–	–	497	–
TeaCache (fast) [19]	0.7604	0.2913	16.17	0.5685	117.61	249	2.00×
Ours	**0.7615**	**0.1256**	**22.02**	**0.7899**	**62.56**	**247**	**2.01**×

Table 2. Quantitative comparison with prior methods under Wan2.1 baselines.

steps $\mathcal{S}_{\text{comp}}$ and cache utilization steps $\mathcal{S}_{\text{use}}$, where:

$$\mathcal{S}_{\text{comp}} = \{s \mid s \bmod T_s = 0, s > s_0\} \tag{12}$$

During cache computation steps ($s \in \mathcal{S}_{\text{comp}}$), the hidden state update follows:

$$h_s = h_s + \Delta^{(i)}_{\text{cache}}, \tag{13}$$

where $\Delta^{(i)}_{\text{cache}}$ is computed according to our block-level strategy. For cache utilization steps ($s \in \mathcal{S}_{\text{use}}$), we directly reuse cached features for background blocks while maintaining full computation for foreground blocks:

$$h_s = \begin{cases} h_s + \Delta^{(i)}_{\text{cache}}, & \text{if } i \in \mathcal{B}, \\ \text{Block}_i(h_s), & \text{if } i \in \mathcal{F}. \end{cases} \tag{14}$$

To effectively balance computational efficiency with generation quality, we implement an adaptive caching interval:

$$T_s = T_{\max} - (T_{\max} - T_{\min}) \cdot \frac{s - s_0}{S - s_0}, \tag{15}$$

where $T_{\max}$ and $T_{\min}$ represent the initial and final caching intervals, S denotes the total diffusion steps, and s_0 indicates the warm-up threshold. This progressive adjustment of caching frequency enables more frequent feature reuse as the diffusion process stabilizes, while preserving computational precision during critical early steps.

4. Experiment

4.1. Implementation Details

Comparison experiments for HunyuanVideo are completed on NVIDIA H200 140GB GPUs using PyTorch. FlashAttention [7] is enabled by default for all experiments. The default parameters follow the official HunyuanVideo settings: 720×1280 resolution, 129 frames, 50 inference steps, and a fixed seed. Further experiments on Wan2.1 [31] is carried on H100 80GB GPUs, the hardware setting between baselines and ours are the same.

4.2. Metrics

To evaluate the performance of video synthesis acceleration methods, we focus on two key aspects: inference efficiency and visual quality. For inference efficiency, we use Floating Point Operations (FLOPs) and inference latency as our primary metrics. We employ VBench [13], LPIPS [38], PSNR, and SSIM for visual quality assessment. VBench [13] is a comprehensive benchmark suite for video generative models, aligning well with human perception and providing valuable insights from multiple perspectives. Additionally, LPIPS, PSNR, and SSIM measure the similarity between videos generated by the accelerated sampling method and those from the original model: PSNR (Peak Signal-to-Noise Ratio) evaluates pixel-level fidelity, LPIPS (Learned Perceptual Image Patch Similarity) measures perceptual consistency, and SSIM (Structural Similarity Index) assesses structural similarity.

4.3. Comparison Experiment

We evaluate our proposed method with three baselines: HunyuanVideo [15], TeaCache-slow [19], and TeaCache-fast [19], using six performance metrics: VBench, LPIPS, PSNR, SSIM, FID, and latency.

Tab. 1 and Tab. 2 present the results. The HunyuanVideo serves as the baseline with a VBench score of 0.7703 and a latency of 1745 ms. **TeaCache-slow** shows a marginal

Method	Visual Quality Evaluation					Efficiency Evaluation	
	VBench ↑	LPIPS ↓	PSNR ↑	SSIM ↑	FID ↓	Latency (ms) ↓	Speedup ↑
HunyuanVideo (Baseline)	**0.7703**	–	–	–	–	1745	–
Foreground-focused Block Reuse	0.7685	0.1449	24.23	0.8065	58.53	1113	1.57×
Background-focused Block Reuse	0.7642	**0.1203**	**26.44**	**0.8445**	**41.10**	932	**1.87**×
Background+Foreground (Split)	0.7637	0.1753	23.83	0.7964	61.79	1056	1.65×
Background+Foreground (Alternate)	0.7623	0.1875	23.46	0.7893	55.89	**1056**	1.65×

Table 3. Ablation on block-level reuse strategies. Foreground reuse achieves the best perceptual quality and distortion metrics. "Split" alternates block types across step groups, while "Alternate" switches reuse types between steps.

Method	Visual Quality Evaluation					Efficiency Evaluation	
	VBench ↑	LPIPS ↓	PSNR ↑	SSIM ↑	FID ↓	Latency (ms) ↓	Speedup ↑
HunyuanVideo (Baseline)	**0.7703**	–	–	–	–	1745	–
Foreground-focused Block Reuse							
Foreground+Stepwise	0.7685	0.1449	24.23	0.8065	58.53	1113	1.57×
Foreground+Step Inverse	0.7681	0.1785	24.80	0.7735	65.57	1098	1.59×
Foreground+Step Average	0.7698	0.1303	25.10	0.8193	50.29	1089	1.60×
Background-focused Block Reuse							
Background+Stepwise	0.7642	**0.1203**	**26.44**	**0.8445**	**41.10**	932	**1.87**×
Background+Step Inverse	0.7517	0.2529	24.49	0.7495	41.10	934	1.87×
Background+Step Average	0.7627	0.1571	25.22	0.8275	53.20	**924**	1.89×

Table 4. Ablation on step-level reuse strategies across background and foreground blocks. "Stepwise" denotes decreasing reuse intervals; "Step Inverse" denotes increasing reuse intervals; "Step Average" denotes fixed reuse intervals.

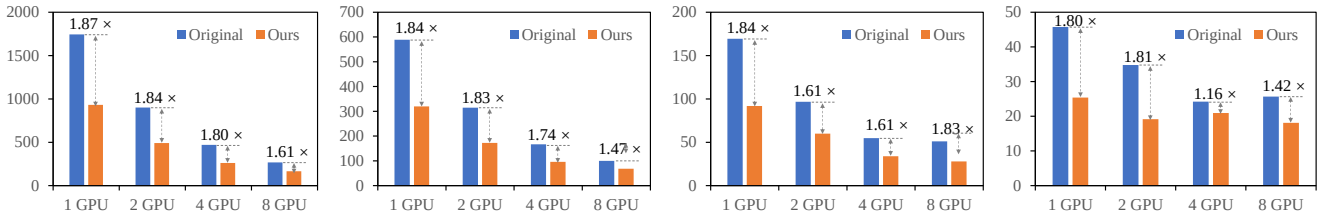
decrease in VBench (0.76998) while achieving moderate visual quality (LPIPS = 0.1720, PSNR = 21.91, SSIM = 0.7456) with a significantly reduced latency of 1052 ms (1.66 faster than the baseline). **TeaCache-fast** further lowers latency to 753 ms (2.31 faster than the baseline) at the cost of degraded image quality, reflected in higher LPIPS (0.1830) and FID (83.85). Our approach achieves notable improvements in reconstruction metrics: LPIPS is reduced to 0.1203, PSNR increases to 26.44, SSIM improves to 0.8445, and FID decreases to 41.10, indicating superior perceptual quality compared to the TeaCache variants. Additionally, our method achieves a latency of 932 ms (1.87 faster than the baseline), demonstrating efficiency gains without compromising performance.

4.4. Ablation Study

We conduct ablation studies to investigate the impact of reusing background and foreground information at different stages on both visual quality and runtime performance.

Ablation on Block-Level Reuse. We first investigate the impact of reusing different types of blocks—**Foreground-focused** versus **Background-focused**. The results are shown in Tab. 3. Our analysis shows that **Background-focused block reuse** (*i.e.*, reusing blocks that focus on background content) helps preserve global consistency and yields a competitive VBench score (0.768), but struggles with fine-grained visual quality (LPIPS = 0.1449, PSNR = 24.23, FID = 58.53). In contrast, **Foreground-focused block reuse** (*i.e.*, reusing blocks that focus on dynamic foreground content) is more effective for detailed reconstruction, achieving substantially better LPIPS (0.1203), PSNR (26.44), and SSIM (0.8445), along with a lower FID of 41.10. These results suggest that background features are more spatially localized and temporally variant, thus benefiting more from selective reuse.

Ablation on Reuse Patterns. We further compare different reuse scheduling strategies across block types, as shown in Tab. 3. In the *"Split"* pattern, foreground-focused blocks are reused in early steps and background-focused blocks in later ones. In contrast, *"Alternate"* reuse switches between them at each step segment. Split reuse provides stronger consistency across frames by stabilizing background modeling, while alternate reuse improves adaptability by interleaving static and dynamic feature attention. Notably, both



(a) 720p, 129 frames (b) 720p, 69 frames (c) 360p, 129 frames (d) 360p, 49 frames

Figure 7. Inference efficiency of our method at different resolutions and video lengths.

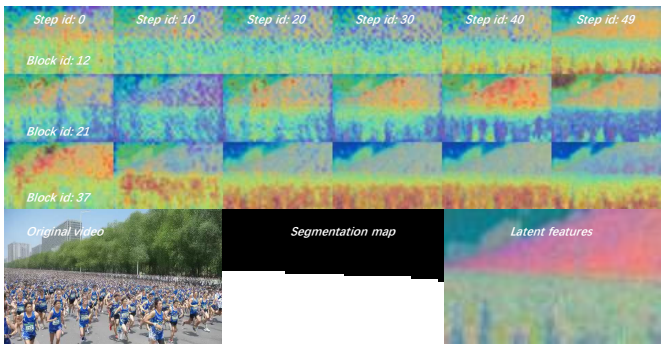


Figure 8. Attention map visualization for a multi-object video.

are outperformed by the **Background-focused block reuse** strategy (*i.e.*, reusing background-focused blocks with decreasing intervals), which aligns the reuse pattern with the generation trajectory. This method best exploits attention specialization across blocks, achieving the best trade-off between distortion (PSNR, SSIM) and perceptual realism (FID, LPIPS) with modest latency.

Ablation on Step-Level Reuse. To explore how reuse frequency affects performance, we vary the reuse interval, defined as the number of steps a cached block is retained before recomputation. As shown in Tab. 4, a *"Stepwise"* strategy—where reuse intervals gradually decrease—consistently improves perceptual quality over fixed (*"Step Average"*) or increasing (*"Step Inverse"*) schedules. For instance, the **Background+Stepwise** strategy outperforms its inverse counterpart in both PSNR (26.44 vs. 24.49) and LPIPS (0.1203 vs. 0.2529), while being nearly twice as fast as the baseline (932 ms vs. 1745 ms). These findings highlight that reusing stable (background-focused) blocks early, followed by more frequent recomputation of dynamic (foreground-focused) blocks later, helps avoid stale information accumulation while maintaining quality.

4.5. Ablation about resolution and GPU numbers

We further evaluate our method at different resolutions (720p and 360p) and frame counts (129, 69, and 49 frames), using 1, 2, 4, and 8 GPUs to measure speedup over the original method. As shown in Fig. 7, our approach consistently outperforms the baseline. At 720p with 129 frames, we achieve a 1.87× speedup on a single GPU, maintaining 1.84× with 2 GPUs and 1.80× with 4 GPUs. However, with 8 GPUs, the speedup drops to 1.61× due to communication overhead. A similar pattern is observed for shorter videos, with diminishing efficiency as GPU count increases. These results demonstrate that our method effectively reduces computational time, particularly for high-resolution and long-duration, while maintaining robust scalability across multiple GPUs.

4.6. Discussion on Multiple Objects

To investigate the applicability of our attentional analysis in multi-object scenarios, we visualize the attention for a video with the prompt *"a marathon with tens of runners"*, as shown in Fig. 8. Despite the presence of numerous objects, their significant motion and prominence categorize them as foreground, resulting in higher attention scores. The results demonstrate that the model maintains its ability to distinguish between background and foreground using specific blocks (*e.g.*, blocks 12, 21, and 37), consistent with the single-object case.

5. Conclusion

We conduct a systematic analysis of attention dynamics in the DiT framework and propose ProfilingDiT, a semantically guided caching strategy. Leveraging foreground-background segmentation and attention distribution priors, ProfilingDiT adaptively caches informative blocks and adjusts caching frequency across denoising steps. Experiments demonstrate that ProfilingDiT achieves significant acceleration with minimal degradation in video quality, highlighting its effectiveness for optimizing DiT-based diffusion models.

Future Work. We plan to extend and deploy it across a broader range of video diffusion models to prove its generalizability and further enhance its robustness in diverse scenarios.

References

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010. 4

[2] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025. 2

[3] Harold Haodong Chen, Haojian Huang, Xianfeng Wu, Yexin Liu, Yajing Bai, Wen-Jie Shu, Harry Yang, and Ser-Nam Lim. Temporal regularization makes your video generator stronger. *arXiv preprint arXiv:2503.15417*, 2025. 2

[4] 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, 3

[5] 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. 3, 5

[6] Shoufa Chen, Chongjian Ge, Yuqi Zhang, Yida Zhang, Fengda Zhu, Hao Yang, Hongxiang Hao, Hui Wu, Zhichao Lai, Yifei Hu, et al. Goku: Flow based video generative foundation models. *arXiv preprint arXiv:2502.04896*, 2025. 2

[7] 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

[8] Yuexi Dong. Enhancing painting exhibition experiences with the application of augmented reality-based ai video generation technology. In *International Conference on Human-Computer Interaction*, pages 256–262. Springer, 2025. 2

[9] Jiasong Feng, Ao Ma, Jing Wang, Bo Cheng, Xiaodan Liang, Dawei Leng, and Yuhui Yin. Fancyvideo: Towards dynamic and consistent video generation via cross-frame textual guidance. *arXiv preprint arXiv:2408.08189*, 2024. 2

[10] Yingqing He, Menghan Xia, Haoxin Chen, Xiaodong Cun, Yuan Gong, Jinbo Xing, Yong Zhang, Xintao Wang, Chao Weng, Ying Shan, et al. Animate-a-story: Storytelling with retrieval-augmented video generation. *arXiv preprint arXiv:2307.06940*, 2023. 2

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 2

[12] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022. 2

[13] Ziqi Huang, Yinan 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

[14] 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, 2024. 2, 3

[15] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024. 1, 2, 3, 4, 6

[16] 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, 2023. 2, 3

[17] Bin Lin, Yunyang Ge, Xinhua Cheng, Zongjian Li, Bin Zhu, Shaodong Wang, Xianyi He, Yang Ye, Shenghai Yuan, Liuhan Chen, et al. Open-sora plan: Open-source large video generation model. *arXiv preprint arXiv:2412.00131*, 2024. 2

[18] Shanchuan Lin, Xin Xia, Yuxi Ren, Ceyuan Yang, Xuefeng Xiao, and Lu Jiang. Diffusion adversarial post-training for one-step video generation, 2025. 2, 3

[19] Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. *arXiv preprint arXiv:2411.19108*, 2024. 2, 3, 6

[20] Haozhe Liu, Wentian Zhang, Jinheng Xie, Francesco Faccio, Mengmeng Xu, Tao Xiang, Mike Zheng Shou, Juan-Manuel Perez-Rua, and Jürgen Schmidhuber. Faster diffusion via temporal attention decomposition. *arXiv preprint arXiv:2404.02747*, 2024. 2, 3

[21] Yexin Liu and Lin Wang. Mycloth: Towards intelligent and interactive online t-shirt customization based on user's preference. In *2024 IEEE Conference on Artificial Intelligence (CAI)*, pages 955–962. IEEE, 2024. 2

[22] 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, 2022. 2

[23] 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

[24] Guoqing Ma, Haoyang Huang, Kun Yan, Liangyu Chen, Nan Duan, Shengming Yin, Changyi Wan, Ranchen Ming, Xiaoniu Song, Xing Chen, Yu Zhou, Deshan Sun, Deyu Zhou, Jian Zhou, Kaijun Tan, Kang An, Mei Chen, Wei Ji, Qiling Wu, Wen Sun, Xin Han, Yanan Wei, Zheng Ge, Aojie Li, Bin Wang, Bizhu Huang, Bo Wang, Brian Li, Changxing Miao, Chen Xu, Chenfei Wu, Chenguang Yu, Dapeng Shi, Dingyuan Hu, Enle Liu, Gang Yu, Ge Yang, Guanzhe Huang, Gulin Yan, Haiyang Feng, Hao Nie, Haonan Jia, Hanpeng Hu, Hanqi Chen, Haolong Yan, Heng Wang, Hongcheng Guo, Huilin Xiong, Huixin Xiong, Jiahao Gong, Jianchang Wu, Jiaoren Wu, Jie Wu, Jie Yang, Jiashuai Liu, Jiashuo Li, Jingyang Zhang, Junjing Guo, Junzhe Lin, Kaixiang Li, Lei Liu, Lei Xia, Liang Zhao, Liguo Tan, Liwen Huang, Liying Shi, Ming Li, Mingliang Li, Muhua Cheng, Na Wang,

Qiaohui Chen, Qinglin He, Qiuyan Liang, Quan Sun, Ran Sun, Rui Wang, Shaoliang Pang, Shiliang Yang, Sitong Liu, Siqi Liu, Shuli Gao, Tiancheng Cao, Tianyu Wang, Weipeng Ming, Wenqing He, Xu Zhao, Xuelin Zhang, Xianfang Zeng, Xiaojia Liu, Xuan Yang, Yaqi Dai, Yanbo Yu, Yang Li, Yineng Deng, Yingming Wang, Yilei Wang, Yuanwei Lu, Yu Chen, Yu Luo, Yuchu Luo, Yuhe Yin, Yuheng Feng, Yuxiang Yang, Zecheng Tang, Zekai Zhang, Zidong Yang, Binxing Jiao, Jiansheng Chen, Jing Li, Shuchang Zhou, Xiangyu Zhang, Xinhao Zhang, Yibo Zhu, Heung-Yeung Shum, and Daxin Jiang. Step-video-t2v technical report: The practice, challenges, and future of video foundation model, 2025. 2

[25] Yatian Pang, Bin Zhu, Bin Lin, Mingzhe Zheng, Francis E. H. Tay, Ser-Nam Lim, Harry Yang, and Li Yuan. Dreamdance: Animating human images by enriching 3d geometry cues from 2d poses. *arXiv preprint arXiv: 2412.00397*, 2024. 2

[26] 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. 2

[27] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, David Yan, Dhruv Choudhary, Dingkang Wang, Geet Sethi, Guan Pang, Haoyu Ma, Ishan Misra, Ji Hou, Jialiang Wang, Kiran Jagadeesh, Kunpeng Li, Luxin Zhang, Mannat Singh, Mary Williamson, Matt Le, Matthew Yu, Mitesh Kumar Singh, Peizhao Zhang, Peter Vajda, Quentin Duval, Rohit Girdhar, Roshan Sumbaly, Sai Saketh Rambhatla, Sam Tsai, Samaneh Azadi, Samyak Datta, Sanyuan Chen, Sean Bell, Sharadh Ramaswamy, Shelly Sheynin, Siddharth Bhattacharya, Simran Motwani, Tao Xu, Tianhe Li, Tingbo Hou, Wei-Ning Hsu, Xi Yin, Xiaoliang Dai, Yaniv Taigman, Yaqiao Luo, Yen-Cheng Liu, Yi-Chiao Wu, Yue Zhao, Yuval Kirstain, Zecheng He, Zijian He, Albert Pumarola, Ali Thabet, Artsiom Sanakoyeu, Arun Mallya, Baishan Guo, Boris Araya, Breena Kerr, Carleigh Wood, Ce Liu, Cen Peng, Dimitry Vengertsev, Edgar Schonfeld, Elliot Blanchard, Felix Juefei-Xu, Fraylie Nord, Jeff Liang, John Hoffman, Jonas Kohler, Kaolin Fire, Karthik Sivakumar, Lawrence Chen, Licheng Yu, Luya Gao, Markos Georgopoulos, Rashel Moritz, Sara K. Sampson, Shikai Li, Simone Parmeggiani, Steve Fine, Tara Fowler, Vladan Petrovic, and Yuming Du. Movie gen: A cast of media foundation models, 2025. 2

[28] Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. Fora: Fast-forward caching in diffusion transformer acceleration, 2024. 2, 3

[29] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022. 2

[30] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. 2

[31] Wan Team. Wan: Open and advanced large-scale video generative models. 2025. 2, 4, 6

[32] Fanyi Wang, Peng Liu, Haotian Hu, Dan Meng, Jingwen Su, Jinjin Xu, Yanhao Zhang, Xiaoming Ren, and Zhiwang Zhang. Loopanimate: Loopable salient object animation. In *Proceedings of the 6th ACM International Conference on Multimedia in Asia*, pages 1–8, 2024. 2

[33] Xianfeng Wu, Yajing Bai, Haoze Zheng, Harold Haodong Chen, Yexin Liu, Zihao Wang, Xuran Ma, Wen-Jie Shu, Xianzu Wu, Harry Yang, et al. Lightgen: Efficient image generation through knowledge distillation and direct preference optimization. *arXiv preprint arXiv:2503.08619*, 2025. 2

[34] Jinbo Xing, Long Mai, Cusuh Ham, Jiahui Huang, Aniruddha Mahapatra, Chi-Wing Fu, Tien-Tsin Wong, and Feng Liu. Motioncanvas: Cinematic shot design with controllable image-to-video generation. *arXiv preprint arXiv:2502.04299*, 2025. 2

[35] 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. 2

[36] Juan Zhang, Jiahao Chen, Cheng Wang, Zhiwang Yu, Tangquan Qi, Can Liu, and Di Wu. Virbo: Multimodal multilingual avatar video generation in digital marketing. *arXiv preprint arXiv:2403.11700*, 2024. 2

[37] Lei Zhang, Ximing Wu, Feng Wang, Andy Sun, Laizhong Cui, and Jiangchuan Liu. Edge-based video stream generation for multi-party mobile augmented reality. *IEEE Transactions on Mobile Computing*, 23(1):409–422, 2022. 2

[38] 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

[39] Canyu Zhao, Mingyu Liu, Wen Wang, Weihua Chen, Fan Wang, Hao Chen, Bo Zhang, and Chunhua Shen. Moviedreamer: Hierarchical generation for coherent long visual sequence. *arXiv preprint arXiv:2407.16655*, 2024. 2

[40] Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. *arXiv preprint arXiv:2408.12588*, 2024. 2, 3

[41] Mingzhe Zheng, Yongqi Xu, Haojian Huang, Xuran Ma, Yexin Liu, Wenjie Shu, Yatian Pang, Feilong Tang, Qifeng Chen, Harry Yang, et al. Videogen-of-thought: A collaborative framework for multi-shot video generation. *arXiv preprint arXiv:2412.02259*, 2024. 2

[42] 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. 2

[43] Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching, 2025. 3