

# SortBlock: Similarity-Aware Feature Reuse for Diffusion Model

Hanqi Chen<sup>1\*</sup>, Xu Zhang<sup>2\*</sup>, Xiaoliu Guan<sup>3</sup>, Lielin Jiang<sup>2</sup>, Guanzhong Wang<sup>2†</sup>, Zeyu Chen<sup>2</sup>, Yi Liu<sup>2</sup>

<sup>1</sup>International Joint Innovation Center, The Electromagnetics Academy at Zhejiang University

<sup>2</sup>PaddlePaddle Team, Baidu Inc.

<sup>3</sup>School of Computer Science, Wuhan University

## Abstract

Diffusion Transformers (DiTs) have demonstrated remarkable generative capabilities, particularly benefiting from Transformer architectures that enhance visual and artistic fidelity. However, their inherently sequential denoising process results in high inference latency, limiting their deployment in real-time scenarios. Existing training-free acceleration approaches typically reuse intermediate features at fixed timesteps or layers, overlooking the evolving semantic focus across denoising stages and Transformer blocks. To address this, we propose SortBlock, a training-free inference acceleration framework that dynamically caches block-wise features based on their similarity across adjacent timesteps. By ranking the evolution of residuals, SortBlock adaptively determines a recomputation ratio, selectively skipping redundant computations while preserving generation quality. Furthermore, we incorporate a lightweight linear prediction mechanism to reduce accumulated errors in skipped blocks. Extensive experiments across various tasks and DiT architectures demonstrate that SortBlock achieves over 2 times inference speedup with minimal degradation in output quality, offering an effective and generalizable solution for accelerating diffusion-based generative models.

**Project Page** — <https://sortblock.github.io>

## Introduction

Diffusion models have been widely recognized as a key breakthrough in image and video generation, owing to their powerful generative capabilities. Recently, there has been growing interest in shifting the architecture of diffusion models from U-Net to Transformer-based designs such as Diffusion Transformers (DiTs) (Peebles and Xie 2023). These refined architectures not only produce visually and artistically compelling outputs but also demonstrate superior scaling behavior.

Despite their impressive performance, DiTs remain constrained by slow inference speeds due to their inherently sequential denoising process, limiting their applicability in

real-time scenarios. The core bottleneck stems from the iterative nature of the reverse process, which hinders parallel decoding. Moreover, as model sizes grow and higher-resolution or longer-duration video generation becomes more desirable, the inference latency becomes even more problematic.

To accelerate the visual generation process, prior work has explored distillation-based approaches (Ge et al. 2025; Yin et al. 2024b,a) and post-training techniques. However, these methods typically require additional training, incurring substantial computational costs and data demands. An alternative line of research focuses on caching mechanisms, which aim to accelerate inference without retraining. Such methods leverage the observation that model outputs across adjacent timesteps during denoising tend to be similar, enabling unified caching strategies to reduce redundant computation. Some approaches further propose predictive reuse to improve accuracy over direct caching, while others investigate output patterns across blocks to devise reuse policies.

In this work, we aim to accelerate DiTs by minimizing redundant computation, a plug-and-play paradigm that can generalize across architectures and tasks. While feature redundancy has been widely acknowledged in visual tasks, and recent works (Xie et al. 2025, 2024) have observed its presence in diffusion denoising, redundancy within DiTs and how to mitigate it remains underexplored.

To address this gap, we revisit the feature distances between DiT blocks (as shown in Fig. 2), and propose a training-free acceleration method called SortBlock, which predicts block-level changes in the next timestep and exploits high-similarity transitions to avoid redundant computation. Prior reuse methods (Yu et al. 2025; Shen et al. 2025) often adopt uniform strategies without accounting for multi-scale or varying semantic similarities. Excessive reuse can introduce compounding errors, leading to distorted structures and misalignment with prompts.

In the denoising process, structural content is typically formed at early stages with higher noise levels, while textures and details emerge at later stages with lower noise levels. Moreover, caching deeper (later) DiT blocks tends to favor contour generation, whereas caching shallower (earlier) blocks is more beneficial for preserving fine-grained details. Therefore, applying a static reuse policy derived from local patterns to the entire process may be too arbitrary. We argue

\*Equal Contribution

†Corresponding Author

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

that designing stage-specific reuse strategies is both more principled and more effective in capturing redundancy.

SortBlock divides the denoising process into multiple strategy intervals. For each interval, it ranks DiT blocks based on the similarity of residual changes between the first two timesteps and selectively recomputes blocks with the lowest similarity. To address the issue of growing errors from direct feature reuse, we further introduce a simple effective linear extrapolation mechanism that predicts the next-step feature and reduces reconstruction error with negligible overhead. Despite its simplicity, this strategy enables more aggressive caching while maintaining generation quality.

We evaluate SortBlock on the Flux.1-dev(Labs 2024), Wan2.1(WanTeam et al. 2025) and HunyuanVideo(Kong et al. 2025) baseline. Experimental results show that our training-free method achieves over 2× inference speedup without sacrificing perceptual quality. Moreover, SortBlock consistently outperforms other training-free acceleration baselines.

## Related Work

**Diffusion Transformer.** Diffusion models have emerged as key players in the generative modeling landscape due to their impressive capabilities. Traditionally, U-Net-based diffusion models (Zhang et al. 2025b; Rombach et al. 2022) have demonstrated strong performance across various applications, including image and video generation (WanTeam et al. 2025; Ma et al. 2025). Recently, a growing body of work has transitioned to Transformer-based architectures. This architectural shift has led to significant improvements in generating visually and artistically compelling content, while also better adhering to scaling laws. Moreover, DiTs show promising potential in effectively integrating and generating multimodal content. However, their real-time applicability remains limited by the inherently iterative nature of the diffusion process.

**Acceleration of Diffusion Models.** To accelerate the inference process of diffusion models, extensive efforts have been made, which can generally be categorized into two paradigms: reducing the number of sampling steps and reducing the per-step computational cost. The former typically involves designing faster samplers (Song, Meng, and Ermon 2022; Lu et al. 2022; Zhang et al. 2024; Zhao et al. 2023), or performing step-wise distillation (Yin et al. 2024b; Meng et al. 2023). The latter focuses on model-level optimization techniques such as distillation (Ge et al. 2025; Yin et al. 2024a; Starodubcev et al. 2025), pruning (Xie et al. 2025; Fang, Ma, and Wang 2023), or reducing redundant computations (Liu et al. 2025d; Ma, Fang, and Wang 2023; Zhang et al. 2025a; Liu et al. 2025a; Yu et al. 2025; Lv et al. 2025). Some prior studies have identified feature redundancy in U-Net-based diffusion models (Chen et al. 2024), however, their coarse-grained feature reuse strategies often include low-similarity features, leading to structural distortions and misalignment between text and generated images. In contrast, we investigate feature redundancy specifically in DiTs and propose reusing structurally similar spatiotemporal fea-

tures to achieve acceleration while maintaining high consistency with the baseline model outputs.

## Methodology

### Preliminaries

**Forward and Reverse Process in Diffusion Models.** Diffusion models gradually inject noise into data and learn to reverse this process in order to generate clean data from noise. In this work, we focus on the latent-space formulation of noise injection and denoising introduced by (Rombach et al. 2022), where the diffusion process is performed in the latent space. In the forward process, the posterior distribution of the noisy latent variable  $z_t$  at timestep  $t$  admits a closed-form expression:

$$q(z_t|z_0) = \mathcal{N}(z_t; \sqrt{\alpha_t}z_0, (1 - \alpha_t)I), \quad (1)$$

the noise variance sequence is defined as  $\alpha_t = \prod_{i=0}^t (1 - \beta_i)$ , where  $\beta_i \in (0, 1)$ . The reverse process, which entails generating data from noise, constitutes a pivotal component within the diffusion model framework. Upon training the diffusion model  $\epsilon_\theta(z_t, t)$ , the conventional sampler DDPM progressively denoises  $z_t \sim \mathcal{N}(0, I)$  across  $T$  steps. Alternatively, faster samplers akin to DDIM can expedite this procedure utilizing the subsequent formula:

$$\begin{aligned} z_{t-1} = & \sqrt{\alpha_{t-1}} \left( \frac{z_t - \sqrt{1 - \alpha_t} \epsilon_\theta(z_t, t)}{\sqrt{\alpha_t}} \right) \\ & + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \cdot \epsilon_\theta(z_t, t) + \sigma_t \epsilon_t. \end{aligned} \quad (2)$$

During this denoising phase, the model initially generates the coarse structure of the image and subsequently refines it by incorporating textures and detailed information.

**Features In The Transformer.** The number of denoising steps directly correlates with the number of network forward passes in the DiT architecture, which is typically composed of multiple stacked blocks. Each block sequentially computes its output based on the input from the previous one. The shallow blocks, being closer to the input, tend to capture global structures and coarse layouts of the data. In contrast, deeper blocks, located closer to the output, progressively refine fine details, contributing to outputs that are both realistic and visually compelling (Park and Kim 2022; Wu et al. 2021).

### Analysis

To investigate the correlation between DiT blocks and adjacent timesteps, we conduct an in-depth analysis of their behavior throughout the diffusion process.

**Difference with Timesteps.** To better explore variations across timesteps at different denoising stages, we quantify the output differences across all timesteps during the diffusion process, as illustrated in Fig. 1. Our analysis reveals that approximately 70% of the diffusion steps—primarily in the middle region—exhibit low variation in attention outputs. This indicates a high degree of redundancy and serves as the main target for our caching strategy. In contrast, the early and late stages show significantly larger variations,

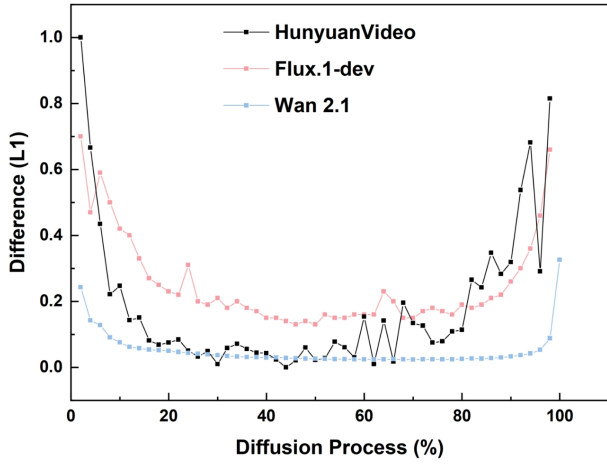


Figure 1: Visualization of output differences in consecutive timesteps of Wan2.1, HunyuanVideo and Flux.1-dev.

suggesting that these regions require higher recomputation ratios—or even full recomputation—in order to minimize the discrepancy between accelerated and original model outputs.

**Difference with Blocks.** In an ideal scenario, if we could obtain in advance the variations of each DiT block at the next time step, we could directly measure the differences between the variations of adjacent time step DiT blocks and decide whether to recompute these DiT blocks based on these differences. In this paper, we use cosine similarity as the metric. For instance, the cosine similarity for the variations of the same DiT block across adjacent time steps is calculated as follows:

$$\mathbf{S}_N = \frac{(\Delta_k^N)^T \Delta_{k+1}^N}{\|\Delta_k^N\| \|\Delta_{k+1}^N\|}, \quad (3)$$

Where  $\Delta_k^N$  represents the input-output variation of the  $N$ -th DiT block at the  $K$ -th step, and  $\Delta_{k+1}^N$  represents the input-output variation of the  $N$ -th DiT block at the  $(K+1)$ -th step. A value of  $\mathbf{S}_N$  closer to 0 indicates greater differences between the two DiT blocks, suggesting they should be recomputed. Conversely, an  $\mathbf{S}_N$  closer to 1 implies higher similarity, allowing for the reuse of information from the previous time step to accelerate model inference speed. Therefore, Eq. 3 serves as the criterion to define and judge whether a DiT block needs to be recomputed.

**Feature Analysis.** However, in most cases, the model’s output cannot be obtained in advance, rendering the aforementioned methods infeasible. To address this issue, an intuitive approach is to design a caching strategy by effectively estimating the input-output changes of the DiT blocks. Taking the Flux.1-dev model as an example, as shown in Fig. 2 (b)(c), the vertical axis represents the L1 distance between the input and output of each block. In the middle region of the dual-stream block and most preceding single-stream blocks, the feature representation exhibits a smooth evolution as the layer depth increases. Based on this insight, we aim to make a simple prediction of the input-output changes

for the DiT blocks in the next time step of the model to satisfy the similarity criterion outlined in Eq. 3.

## Training-Free Acceleration Approach

To alleviate the low inference efficiency problem of DiTs, we introduce SortBlock, a training-free caching framework. Diffusion models operate by gradually denoising images step by step based on time steps, however, this evolutionary process exhibits sparsity because the changes in features between consecutive steps during intermediate denoising are often highly similar. As illustrated in Fig. 2, this sparsity suggests that recalculating all features at every step is generally unnecessary.

To leverage this sparsity, SortBlock selectively updates only a small subset of DiT blocks that exhibit the most significant changes between adjacent steps. The challenge lies in efficiently and accurately identifying these DiT blocks. As shown in Fig. 3, we compute the cosine similarity between the input-output changes of DiT blocks at step  $K$  and step  $K+1$ . By utilizing the cosine similarity between the input-output changes of each DiT block, we identify those DiT blocks with the most substantial changes. Only these selected DiT blocks undergo complete feature recomputation and cache updates.

Calculating the cosine similarity requires features from two adjacent time steps, and full computation of these features would incur high time costs, thus limiting the upper bound of the strategy’s speed. By leveraging the continuous nature of feature trajectories, we employ a predictive caching method that estimates intermediate features across time steps.

**Linear Prediction Method.** Unlike the approach of directly copying features, we cache both the feature values and their time differences:  $C(x_t^N) := \{\mathcal{F}(x_t^N), \Delta\mathcal{F}(x_t^N)\}$ . This enables us to predict the feature trajectory at time  $t-k$  using the formula:

$$\mathcal{F}_{\text{pred}}(x_{t-k}^N) = \mathcal{F}(x_t^N) + \frac{\mathcal{F}(x_t^N) - \mathcal{F}(x_{t+L}^N)}{L}k, \quad (4)$$

where  $\mathcal{F}(x_t^N)$  and  $\mathcal{F}(x_{t+L}^N)$  represent the eigenvalues calculated at the complete time steps  $t$  and  $t+L$ , respectively.  $L$  denotes the time step interval. This first-order approximation effectively captures the linear trend of the feature trajectory, thereby significantly enhancing the accuracy of direct feature reuse. Notably, our feature prediction mechanism is not restricted to calculating subsequent time steps or for similarity computations, instead, it replaces all components that require cached and reused feature values with linear predictions.

The caching strategy is governed by two hyperparameters: the **policy refresh interval**  $K$  and the **self-adaptive update ratio**  $\rho \in [0, 1]$ . The inference process typically involves these components:

**Initialization:** When  $k \equiv 0 \pmod{K}$ , all features are computed, and the input-output changes  $\Delta_k^N$  for all DiT blocks are stored.

**Linear Prediction:** When  $k \equiv 1 \pmod{K}$ , based on the feature values obtained from the previous time step, all fea-

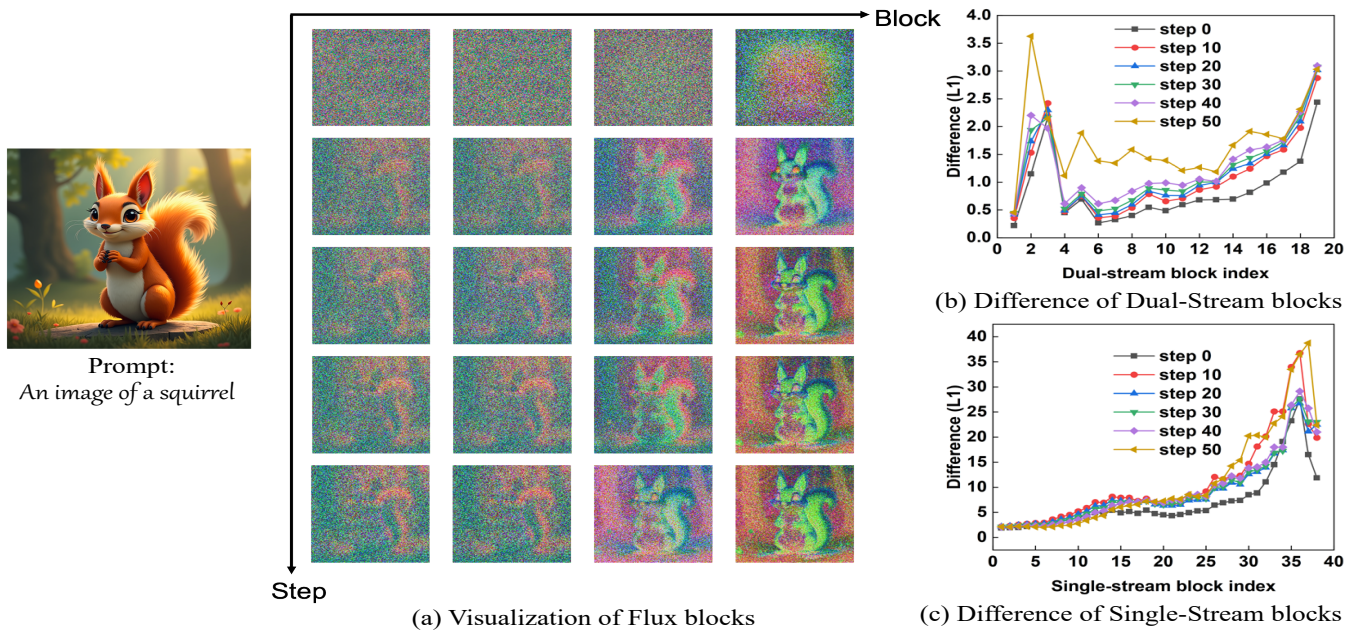


Figure 2: **Visualization of Flux.1-dev blocks.** (a) Visualization of block dynamics as the network depth and timesteps increase. (b) L1 distance variations between dual-stream attention blocks in Flux.1-dev. (c) L1 distance variations between single-stream attention blocks in Flux.1-dev.

tures for the current time step are predicted using Eq. 4, and the input-output changes  $\Delta_{k+1}^N$  for all DiT blocks are stored.

**Identify Changes:** The cosine similarity  $s_N$  between the current input-output changes  $\Delta_{k+1}^N$  and the previous changes  $\Delta_k^N$  is calculated for each DiT block.

**Select DiT Blocks:** The  $[\rho N_{\max}]$  DiT blocks with the lowest  $s_N$  values are selected and marked as 1 in the strategy sequence  $I$ , with the remaining blocks marked as 0.

**Recompute and Predict:** Only the DiT blocks with  $I_{\text{index}} = 1$  are recomputed; the rest are predicted using linear prediction.

**Update Cache:** The cached values are replaced with the newly computed values for the DiT blocks that were recomputed.

**Summary:** This adaptive update mechanism leverages temporal stability to minimize computational cost while maintaining generation quality through selective feature refreshment.

## Experiment

### Settings

**Base Models and Compared Methods.** To demonstrate the effectiveness of our approach, we apply our acceleration technique to both image and video generation models, including Flux.1-dev, Wan2.1 (WanTeam et al. 2025) and HunyuanVideo (Kong et al. 2025). To highlight its advantages, we compare our base models with several recent efficient video synthesis methods, such as PAB (Zhao et al. 2025), T-GATE (Liu et al. 2025b), TaylorSeer (Liu et al. 2025c) and TeaCache (Liu et al. 2025a). Notably, T-GATE

was originally designed for accelerating image synthesis, and PAB adapts it for video generation to enable fair comparison.

**Implementation Detail.** All image generation experiments are conducted on NVIDIA A800 80GB GPUs using the PaddlePaddle framework, while all video generation experiments are performed on NVIDIA H100 80GB GPUs with the same framework. FlashAttention (Dao et al. 2022) is enabled by default in all experiments.

For SortBlock, we apply caching and reuse primarily within the denoising range of 20% to 100% for both Flux.1-dev, Wan2.1 and HunyuanViedo. The recomputation ratio  $\rho$  is defined as a timestep-dependent function, fitted to the curve of L1 distances between adjacent timesteps, as illustrated in Fig. 4. We compare polynomial fits of degree 3 to 5, and empirically choose a 5th-degree curve as the final formulation.

To make  $\rho$  adjustable, we introduce a global scaling hyperparameter  $\beta \in [0, 1]$ , which controls the aggressiveness of the caching policy and allows tuning of the trade-off between speedup and generation quality.

### Main Results

**Accelerate Flux.1-dev for Text-to-Image Generation.** We follow the evaluation protocol established in Flux.1-dev to assess our method on high-resolution text-to-image generation tasks. We report standard metrics including FID (Heusel et al. 2018), CLIP score (Radford et al. 2021), Image Reward (Xu et al. 2023), PSNR and SSIM (Hore and Ziou 2010) to comprehensively evaluate the visual quality and semantic alignment of generated images.

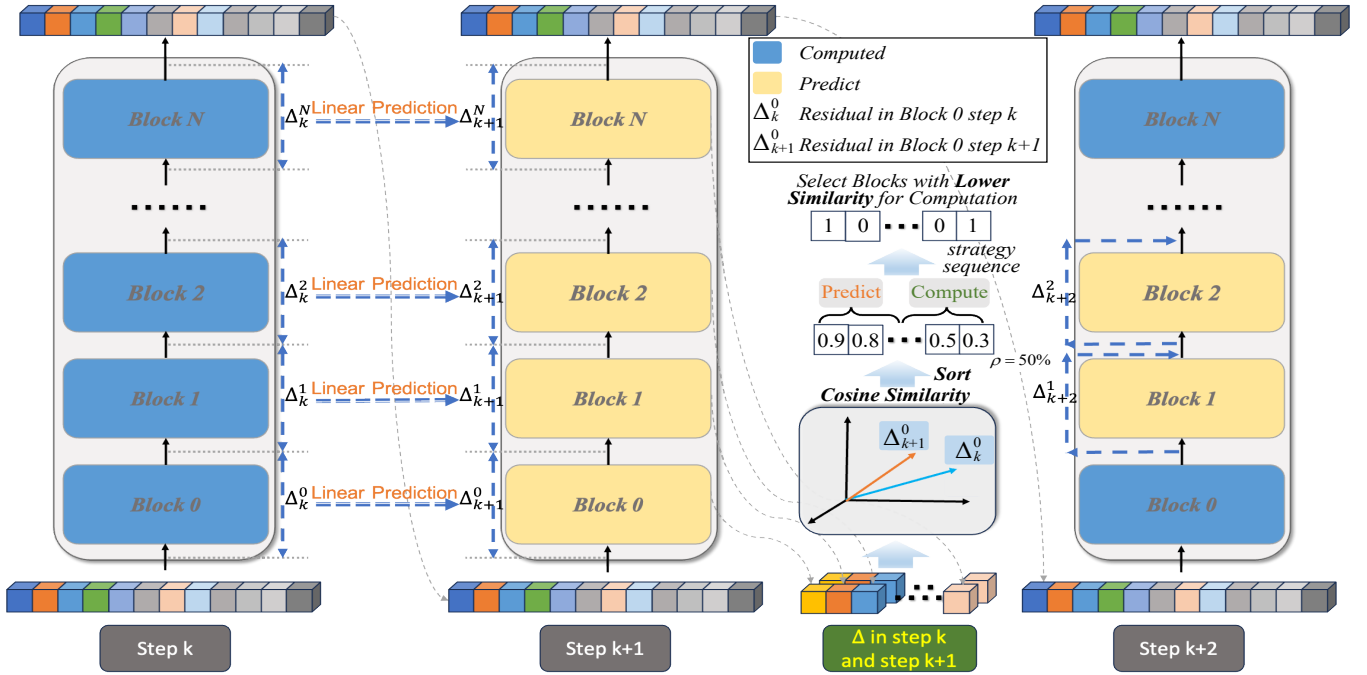


Figure 3: **The SortBlock pipeline.** Block features are adaptively updated based on their similarity between fully computed and predicted counterparts. Specifically, blocks with low similarity are recomputed, while the others are efficiently approximated via prediction.

Method	Speed		Image Quality					
	Speedup $\uparrow$	Latency (s) $\downarrow$	FID $\downarrow$	IR $\uparrow$	CLIP $\uparrow$	SSIM $\uparrow$	PSNR $\uparrow$	LPIPs $\downarrow$
Flux.1-dev, 50 steps	1.0 $\times$	23.23	70.59	1.064	0.332	-	-	-
T-GATE (m=25)	1.2 $\times$	19.44	<b>69.40</b>	1.056	<b>0.332</b>	0.734	19.96	0.271
PAB	1.37 $\times$	16.97	69.72	1.001	0.331	0.849	24.65	0.207
TeaCache (rel=0.25)	1.73 $\times$	13.45	70.14	1.035	0.331	0.805	22.11	0.198
TaylorSeer (N=3 O=3)	2.04 $\times$	11.37	70.27	1.048	0.330	0.763	20.56	0.241
SortBlock (K=5 (900,50))	2.00 $\times$	11.60	70.47	1.067	0.327	<b>0.952</b>	<b>31.49</b>	<b>0.0748</b>
SortBlock (K=9 (900,50))	<b>2.39<math>\times</math></b>	<b>9.7</b>	70.30	<b>1.077</b>	0.328	0.910	27.27	0.1256

Table 1: Text-to-image generation on Flux.1-dev.

Inference efficiency is measured by the generation latency, defined as the average time to generate a single image or video on an NVIDIA A800 GPU. We validate the effectiveness of SortBlock on the COCO dataset. As shown in Table 1, we compare our method against several recent acceleration baselines extended to the Flux.1-dev model, including PAB (Zhao et al. 2025), T-GATE (Liu et al. 2025b), TaylorSeer (Liu et al. 2025c) and TeaCache (Liu et al. 2025a).

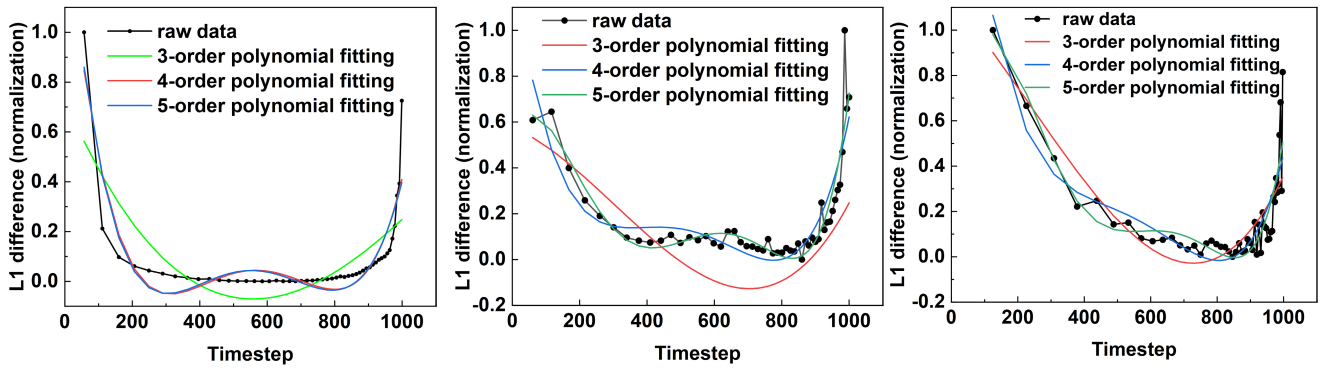
For a fair comparison, we adapt each baseline to the Flux.1-dev backbone. Specifically, for PAB, we reduce the computational overhead by reconfiguring the broadcast mechanism. For T-GATE, the gating value is fixed to 25 throughout the denoising process. In TaylorSeer, we set the reuse factor to 3 and use a polynomial degree of 3.

When the policy refresh interval  $K = 5$  and the policy is applied within timesteps  $\in [50, 900]$ , SortBlock achieves a 2 $\times$  speedup on Flux.1-dev while maintaining comparable

image quality in terms of both visual aesthetics and prompt alignment. Notably, different speed-quality trade-offs can be flexibly controlled by tuning  $K$  and the application interval of the caching strategy.

Compared to T-GATE and PAB, SortBlock achieves superior performance across all image quality metrics while maintaining comparable inference speed. Our method explicitly reduces redundant structural computations during the denoising stages by avoiding recomputation of highly similar block features, thereby minimizing quality degradation.

In contrast, T-GATE, TaylorSeer and PAB apply one or two fixed reuse strategies throughout the entire denoising process without considering the characteristics of different denoising stages. This often leads to suboptimal reuse policies across timesteps, resulting in structural distortions and degraded prompt alignment. TeaCache alleviates de-



(a) Polynomial fitting curves of Wan 2.1 (b) Polynomial fitting curves of Flux-dev (c) Polynomial fitting curves of HunyuanVideo

Figure 4: Correlation visualization between input and output differences across consecutive timesteps in Wan2.1, Flux.1-dev and HunyuanVideo. While raw data points deviate significantly from linearity, polynomial fitting improves the approximation accuracy.

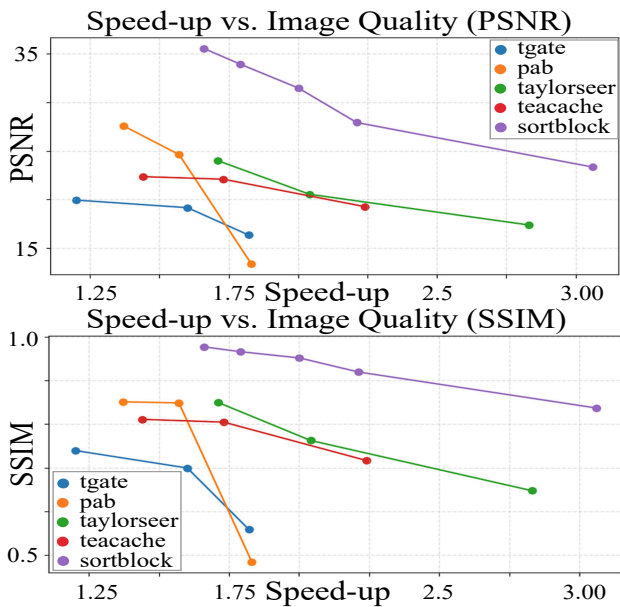


Figure 5: **Quality-latency comparison of cache method with different speed.** PSNR and SSIM comparisons of SortBlock, T-GATE, PAB, TaylorSeer, and TeaCache using Flux.1-dev. SortBlock achieves the best overall balance between visual quality and efficiency.

noising cost through step-wise reuse, but its aggressive reuse can cause unavoidable quality drops. Experimental results demonstrate that under the similar acceleration ratio, SortBlock consistently outperforms these methods across various evaluation metrics.

We further evaluate how the quality of generated images degrades under increasing acceleration ratios across different methods. As shown in Fig. 5, SortBlock consistently preserves high image quality even at higher acceleration levels. Notably, images produced by SortBlock exhibit stronger consistency with the baseline model. This is attributed to

our targeted recomputation strategy, which selectively refines critical features and mitigates quality loss more effectively than uniform reuse approaches.

Method	Speed		Video Quality			
	Speedup $\uparrow$	Lat.(s) $\downarrow$	VBench $\uparrow$	SSIM $\uparrow$	PSNR $\uparrow$	LPIP $\downarrow$
Wan2.1 (50 steps)	1.0 $\times$	86	79.55%	-	-	-
PAB(5,7,9)	1.95 $\times$	44	78.83%	0.878	27.50	0.076
TeaCache(rel=0.08)	1.95 $\times$	44	79.38%	0.903	28.09	0.062
Taylorseer(N=2,O=1)	1.79 $\times$	48	78.94%	0.715	18.75	0.258
SortBlock (K=5)	2.00 $\times$	<b>38</b>	<b>79.47%</b>	<b>0.912</b>	<b>29.79</b>	<b>0.048</b>

Table 2: Text-to-video generation on Wan2.1.

**Accelerate Wan2.1 and HunyuanVideo for Text-to-Video Generation.** Table 2 and Table 3 presents a quantitative evaluation of both efficiency and visual quality on Wan2.1 and HunyuanVideo using the VBench benchmark (Huang et al. 2023). Compared with other training-free acceleration baselines, SortBlock consistently achieves superior performance across various backbone models, sampling schedulers, video resolutions, and sequence lengths.

When evaluating on the Wan2.1 (WanTeam et al. 2025) and HunyuanVideo backbone, SortBlock outperforms PAB (Liu et al. 2025c) and TaylorSeer (Liu et al. 2025c) in all visual quality metrics under comparable acceleration ratios, and even delivers faster inference than PAB while preserving better image fidelity. Although TeaCache (Liu et al. 2025a) demonstrates competitive results, SortBlock still maintains a noticeable advantage in terms of both acceleration and visual quality. These results highlight the generalizability and robustness of our method in diverse settings.

**Qualitative Results.** We further conduct qualitative comparisons to illustrate the visual effects of different acceleration strategies, as shown in Fig. 5. T-GATE introduces a binary gating mechanism to partition the denoising process into early and late stages. However, this rigid division tends to oversimplify the denoising dynamics, often result-

Method	Speed		Video Quality			
	Speedup $\uparrow$	Lat.(s) $\downarrow$	VBench $\uparrow$	SSIM $\uparrow$	PSNR $\uparrow$	LPIP $\downarrow$
HunyuanVideo (50 steps)	1.0 $\times$	53	79.04%	-	-	-
PAB(4,5,6)	1.96 $\times$	27	78.06%	0.71	19.82	0.277
TeaCache(rel=0.15)	2.40 $\times$	22	78.81%	0.72	20.43	0.257
Taylorseer(N=3,O=3)	2.40 $\times$	22	<b>79.14%</b>	0.73	20.98	0.231
SortBlock (K=5)	2.40 $\times$	<b>22</b>	78.29%	<b>0.85</b>	<b>28.29</b>	<b>0.089</b>

Table 3: Text-to-video generation on HunyuanVideo.

ing in blocky artifacts and degraded image quality. Although PAB achieves reasonable visual performance under a 1.37 $\times$  speed-up, it requires manual adaptation for each specific Transformer architecture, making it less practical and time-consuming to deploy.

TeaCache achieves favorable acceleration-quality trade-offs by reusing steps in the denoising process. Nevertheless, its coarse-grained reuse strategy can introduce significant structural deviations and semantic distortions, especially in complex scenes. TaylorSeer leverages Taylor series expansion on MLP and attention modules to predict outputs from previous layers. While it effectively captures local historical patterns, it often lacks broader structural awareness, resulting in suboptimal performance on visual metrics.

In contrast, SortBlock offers a 2 $\times$  acceleration without requiring any additional training. It selectively recomputes block features based on their predicted similarity, preserving both global structure and fine details. This results in higher fidelity to the original image and eliminates common artifacts observed in other methods.

## Discussion

**Ablation Study Impact of Reuse Frequency.** As illustrated in Fig. 6, we visualize how the generated images evolve with increasing values of  $K$ . As the generation time decreases, the main semantic content of the image remains consistent, while the fidelity of fine-grained details gradually degrades. This observation is consistent with the insights presented in Table. 1, which models the dynamic re-computation ratio over timesteps. Varying the value of  $K$  provides a flexible control mechanism for balancing between generation speed and visual quality.

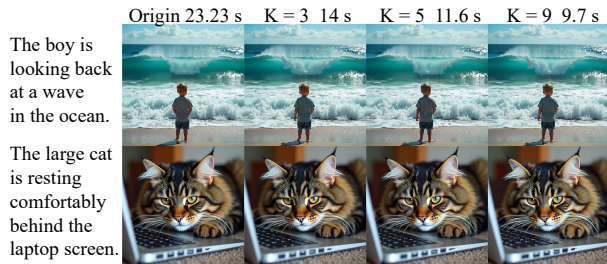


Figure 6: Ablation study on the effect of  $K$ .

**Effect of Using SortBlock at Different Denoising Stages.** As shown in Fig. 7, we investigate the impact of applying SortBlock at different stages of the denoising process:



Figure 7: Effect of using SortBlock at different denoising stages.

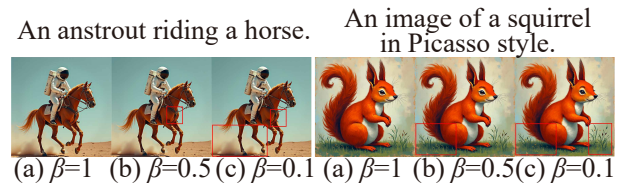


Figure 8: Impact of different  $\beta$ .

the early stage (0%–30%) and the later stage (30%–100%). Since SortBlock primarily reuses blocks with high similarity, applying it during the early stage—where the structural representation is still forming—may introduce structural artifacts or distortions, as highlighted by the red box in Fig. 7 (a). In contrast, applying SortBlock in the later stage, where the global structure is more stable and the focus shifts to refining texture details, leads to more reliable feature reuse and minimal quality degradation, as illustrated in Fig. 7 (b). This demonstrates the effectiveness of our adaptive re-computation strategy across different denoising phases.

**Impact of Reusing Blocks at Different Ratio.** We further investigate the influence of the re-computation ratio by varying the hyperparameter  $\beta$ , as illustrated in Fig. 8. A smaller  $\beta$  results in fewer blocks being re-computed, meaning that features in less similar blocks are predicted without explicit re-computation. This leads to a loss of fine-grained details and texture degradation, as highlighted by the red box in Fig. 8 (b). Conversely, increasing  $\beta$  leads to a higher re-computation ratio, preserving more detailed structures in the generated image but at the cost of increased inference time. Therefore,  $\beta$  serves as a trade-off knob between image quality and generation efficiency.

## Conclusion

In this paper, we introduced SortBlock, an efficient inference acceleration strategy for diffusion models, which adaptively recomputes block features based on their similarity to full-resolution predictions. By skipping redundant computation for similar blocks, SortBlock significantly reduces inference time without introducing additional training. It is applicable to both image and video generation tasks, achieving over 2 $\times$  speed-up while maintaining structural and perceptual fidelity. Extensive experiments demonstrate that SortBlock consistently outperforms other training-free acceleration methods across various backbone models, samplers, and resolutions, offering a strong trade-off between speed and quality.

## Acknowledgements

This work is supported by the National Key Research and Development Program of China under Grant No. 2023YFB4604100.

## References

- Chen, P.; Shen, M.; Ye, P.; Cao, J.; Tu, C.; Bouganis, C.-S.; Zhao, Y.; and Chen, T. 2024.  $\Delta$ -DiT: A Training-Free Acceleration Method Tailored for Diffusion Transformers. arXiv:2406.01125.
- Dao, T.; Fu, D. Y.; Ermon, S.; Rudra, A.; and Ré, C. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. arXiv:2205.14135.
- Fang, G.; Ma, X.; and Wang, X. 2023. Structural Pruning for Diffusion Models. arXiv:2305.10924.
- Ge, X.; Zhang, X.; Xu, T.; Zhang, Y.; Zhang, X.; Wang, Y.; and Zhang, J. 2025. SenseFlow: Scaling Distribution Matching for Flow-based Text-to-Image Distillation. arXiv:2506.00523.
- Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2018. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. arXiv:1706.08500.
- Hore, A.; and Ziou, D. 2010. Image Quality Metrics: PSNR vs. SSIM. In *2010 20th Int. Conf. Pattern Recognit.*, 2366–2369. Istanbul, Turkey: IEEE.
- Huang, Z.; He, Y.; Yu, J.; Zhang, F.; Si, C.; Jiang, Y.; Zhang, Y.; Wu, T.; Jin, Q.; Chanpaisit, N.; Wang, Y.; Chen, X.; Wang, L.; Lin, D.; Qiao, Y.; and Liu, Z. 2023. VBench: Comprehensive Benchmark Suite for Video Generative Models. arXiv:2311.17982.
- Kong, W.; Tian, Q.; Zhang, Z.; Min, R.; Dai, Z.; Zhou, J.; Xiong, J.; Li, X.; Wu, B.; Zhang, J.; Wu, K.; Lin, Q.; Yuan, J.; Long, Y.; Wang, A.; Wang, A.; Li, C.; Huang, D.; Yang, F.; Tan, H.; Wang, H.; Song, J.; Bai, J.; Wu, J.; Xue, J.; Wang, J.; Wang, K.; Liu, M.; Li, P.; Li, S.; Wang, W.; Yu, W.; Deng, X.; Li, Y.; Chen, Y.; Cui, Y.; Peng, Y.; Yu, Z.; He, Z.; Xu, Z.; Zhou, Z.; Xu, Z.; Tao, Y.; Lu, Q.; Liu, S.; Zhou, D.; Wang, H.; Yang, Y.; Wang, D.; Liu, Y.; Jiang, J.; and Zhong, C. 2025. HunyuanVideo: A Systematic Framework For Large Video Generative Models. arXiv:2412.03603.
- Labs, B. F. 2024. FLUX. <https://github.com/black-forest-labs/flux>.
- Liu, F.; Zhang, S.; Wang, X.; Wei, Y.; Qiu, H.; Zhao, Y.; Zhang, Y.; Ye, Q.; and Wan, F. 2025a. Timestep Embedding Tells: It's Time to Cache for Video Diffusion Model. arXiv:2411.19108.
- Liu, H.; Zhang, W.; Xie, J.; Faccio, F.; Xu, M.; Xiang, T.; Shou, M. Z.; Perez-Rua, J.-M.; and Schmidhuber, J. 2025b. Faster Diffusion via Temporal Attention Decomposition. arXiv:2404.02747.
- Liu, J.; Zou, C.; Lyu, Y.; Chen, J.; and Zhang, L. 2025c. From Reusing to Forecasting: Accelerating Diffusion Models with TaylorSeers. arXiv:2503.06923.
- Liu, Z.; Yang, Y.; Zhang, Y.; Chen, J.; Zou, C.; Wei, Q.; Wang, S.; and Zhang, L. 2025d. dLLM-Cache: Accelerating Diffusion Large Language Models with Adaptive Caching. arXiv:2506.06295.
- Lu, C.; Zhou, Y.; Bao, F.; Chen, J.; Li, C.; and Zhu, J. 2022. DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. arXiv:2206.00927.
- Lv, Z.; Si, C.; Song, J.; Yang, Z.; Qiao, Y.; Liu, Z.; and Wong, K.-Y. K. 2025. FasterCache: Training-Free Video Diffusion Model Acceleration with High Quality. arXiv:2410.19355.
- Ma, X.; Fang, G.; and Wang, X. 2023. DeepCache: Accelerating Diffusion Models for Free. arXiv:2312.00858.
- Ma, X.; Wang, Y.; Chen, X.; Jia, G.; Liu, Z.; Li, Y.-F.; Chen, C.; and Qiao, Y. 2025. Latte: Latent Diffusion Transformer for Video Generation. arXiv:2401.03048.
- Meng, C.; Rombach, R.; Gao, R.; Kingma, D. P.; Ermon, S.; Ho, J.; and Salimans, T. 2023. On Distillation of Guided Diffusion Models. arXiv:2210.03142.
- Park, N.; and Kim, S. 2022. How Do Vision Transformers Work? arXiv:2202.06709.
- Peebles, W.; and Xie, S. 2023. Scalable Diffusion Models with Transformers. arXiv:2212.09748.
- Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; and Sutskever, I. 2021. Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020.
- Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. In *2022 IEEE/CVF Conf. Comput. Vis. Pattern Recognit. CVPR*, 10674–10685. New Orleans, LA, USA: IEEE. ISBN 978-1-6654-6946-3.
- Shen, X.; Song, Z.; Zhou, Y.; Chen, B.; Li, Y.; Gong, Y.; Zhang, K.; Tan, H.; Kuen, J.; Ding, H.; Shu, Z.; Niu, W.; Zhao, P.; Wang, Y.; and Gu, J. 2025. LazyDiT: Lazy Learning for the Acceleration of Diffusion Transformers. arXiv:2412.12444.
- Song, J.; Meng, C.; and Ermon, S. 2022. Denoising Diffusion Implicit Models. arXiv:2010.02502.
- Starodubcev, N.; Kuznedelev, D.; Babenko, A.; and Baranchuk, D. 2025. Scale-Wise Distillation of Diffusion Models. arXiv:2503.16397.
- WanTeam; Wang, A.; Ai, B.; Wen, B.; Mao, C.; Xie, C.-W.; Chen, D.; Yu, F.; Zhao, H.; Yang, J.; Zeng, J.; Wang, J.; Zhang, J.; Zhou, J.; Wang, J.; Chen, J.; Zhu, K.; Zhao, K.; Yan, K.; Huang, L.; Feng, M.; Zhang, N.; Li, P.; Wu, P.; Chu, R.; Feng, R.; Zhang, S.; Sun, S.; Fang, T.; Wang, T.; Gui, T.; Weng, T.; Shen, T.; Lin, W.; Wang, W.; Wang, W.; Zhou, W.; Wang, W.; Shen, W.; Yu, W.; Shi, X.; Huang, X.; Xu, X.; Kou, Y.; Lv, Y.; Li, Y.; Liu, Y.; Wang, Y.; Zhang, Y.; Huang, Y.; Li, Y.; Wu, Y.; Liu, Y.; Pan, Y.; Zheng, Y.; Hong, Y.; Shi, Y.; Feng, Y.; Jiang, Z.; Han, Z.; Wu, Z.-F.; and Liu, Z. 2025. Wan: Open and Advanced Large-Scale Video Generative Models. arXiv:2503.20314.

Wu, B.; Xu, C.; Dai, X.; Wan, A.; Zhang, P.; Yan, Z.; Tomizuka, M.; Gonzalez, J.; Keutzer, K.; and Vajda, P. 2021. Visual Transformers: Where Do Transformers Really Belong in Vision Models? In *2021 IEEE CVF Int. Conf. Comput. Vis. ICCV*. Montreal, QC, Canada: IEEE.

Xie, E.; Chen, J.; Chen, J.; Cai, H.; Tang, H.; Lin, Y.; Zhang, Z.; Li, M.; Zhu, L.; Lu, Y.; and Han, S. 2024. SANA: Efficient High-Resolution Image Synthesis with Linear Diffusion Transformers. arXiv:2410.10629.

Xie, E.; Chen, J.; Zhao, Y.; Yu, J.; Zhu, L.; Wu, C.; Lin, Y.; Zhang, Z.; Li, M.; Chen, J.; Cai, H.; Liu, B.; Zhou, D.; and Han, S. 2025. SANA 1.5: Efficient Scaling of Training-Time and Inference-Time Compute in Linear Diffusion Transformer. arXiv:2501.18427.

Xu, J.; Liu, X.; Wu, Y.; Tong, Y.; Li, Q.; Ding, M.; Tang, J.; and Dong, Y. 2023. ImageReward: Learning and Evaluating Human Preferences for Text-to-Image Generation. arXiv:2304.05977.

Yin, T.; Gharbi, M.; Park, T.; Zhang, R.; Shechtman, E.; Durand, F.; and Freeman, W. T. 2024a. Improved Distribution Matching Distillation for Fast Image Synthesis. arXiv:2405.14867.

Yin, T.; Gharbi, M.; Zhang, R.; Shechtman, E.; Durand, F.; Freeman, W. T.; and Park, T. 2024b. One-Step Diffusion with Distribution Matching Distillation. arXiv:2311.18828.

Yu, Z.; Zou, Z.; Shao, G.; Zhang, C.; Xu, S.; Huang, J.; Zhao, F.; Cun, X.; and Zhang, W. 2025. AB-Cache: Training-Free Acceleration of Diffusion Models via Adams-Bashforth Cached Feature Reuse. arXiv:2504.10540.

Zhang, H.; Gao, T.; Shao, J.; and Wu, Z. 2025a. BlockDance: Reuse Structurally Similar Spatio-Temporal Features to Accelerate Diffusion Transformers. arXiv:2503.15927.

Zhang, H.; Wu, Z.; Xing, Z.; Shao, J.; and Jiang, Y.-G. 2024. AdaDiff: Adaptive Step Selection for Fast Diffusion Models. arXiv:2311.14768.

Zhang, T.; Da, C.; Ding, K.; Jin, K.; Li, Y.; Gao, T.; Zhang, D.; Xiang, S.; and Pan, C. 2025b. Diffusion Model as a Noise-Aware Latent Reward Model for Step-Level Preference Optimization. arXiv:2502.01051.

Zhao, W.; Bai, L.; Rao, Y.; Zhou, J.; and Lu, J. 2023. UniPC: A Unified Predictor-Corrector Framework for Fast Sampling of Diffusion Models. arXiv:2302.04867.

Zhao, X.; Jin, X.; Wang, K.; and You, Y. 2025. Real-Time Video Generation with Pyramid Attention Broadcast. arXiv:2408.12588.