

PSA: Pyramid Sparse Attention for Efficient Video Understanding and Generation

Xiaolong Li ^{*1} Youping Gu ^{*1} Xi Lin ^{*1} Weijie Wang ¹ Bohan Zhuang ¹

Abstract

Attention mechanisms are the core of foundation models, but their quadratic complexity remains a critical bottleneck for scaling. This challenge has driven the development of efficient attention mechanisms, with sparsity emerging as the dominant paradigm. Current methods typically retain or discard entire key-value blocks with binary masks, resulting in substantial information loss under high sparsity. To mitigate this gap, we present **Pyramid Sparse Attention (PSA)**, a versatile module applicable to both video understanding and generation tasks. Instead of binary masking, PSA introduces multi-level pooled KV representations, enabling finer mask granularity. Specifically, each query block dynamically allocates lower pooling levels to critical KV blocks and higher levels to less important ones, creating an informative interpolation between full retention and complete pruning. This design, analogous to fixed-point quantization and classical feature pyramid networks in computer vision, effectively mitigates information loss while preserving computational efficiency under a low compute budget. It works with a native, hardware-friendly kernel that leverages decoupled block-tile design to ensure efficient execution. Across video understanding and generation benchmarks, PSA preserves contextual information and visual fidelity, consistently outperforming or achieving comparable performance over existing sparse attention baselines with superior efficiency-quality trade-offs. Our code and model weights are publicly available at: <http://ziplab.co/PSA>

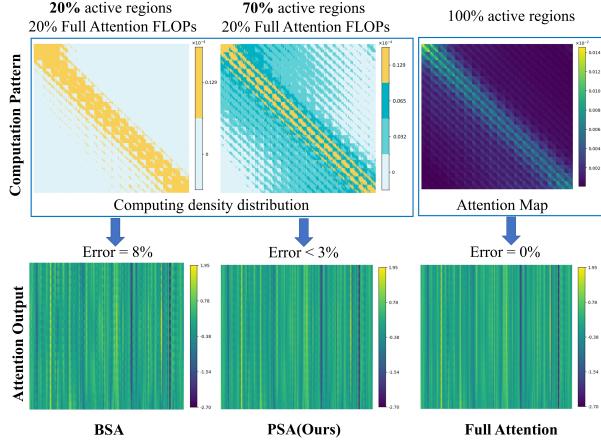


Figure 1. Comparison of attention mechanisms under identical compute budget. All methods use the same input Q , K , and V tensors extracted from Wan2.1–1.3B (Wan et al., 2025) denoising process. Computation Pattern (top-left two panels): Normalized block-wise FLOPs distribution. The two panels plot query blocks on the horizontal axis and key blocks on the vertical axis. Despite identical FLOPs (20% full), the proposed Pyramid Sparse Attention (PSA) allows each query block to attend to a much larger portion of KV blocks (70% active regions), whereas Block Sparse Attention (BSA) (Dao et al., 2022; Zhang et al., 2025a; Xu et al., 2025) restricts each query to only a narrow subset of KV blocks (20% active regions), concentrating FLOPs in limited areas. Attention Output (bottom row): Resulting attention visualizations. PSA closely matches the Full Attention baseline with minimal relative error ($< 3\%$), while BSA shows noticeable distortions due to aggressive pruning.

1. Introduction

Recent advances in *video generation* and *video understanding models* have substantially increased sequence lengths, often reaching tens of thousands of tokens in modern diffusion transformers and autoregressive architectures (Vaswani et al., 2017; Peebles & Xie, 2023; Lin et al., 2024; Wang et al., 2024; Kong et al., 2025; Wan et al., 2025). However, the quadratic complexity of attention has become a major obstacle to serving these models efficiently. In particular, attention computation dominates the prefill stage in long-context video understanding models and the end-to-end sampling process in video generation. For example, when processing high-resolution, long-duration videos, such as generating a

^{*}Equal contribution. ¹ZIP Lab, Zhejiang University. Email: Xiaolong Li <xiaolong.ziplab@gmail.com>, Youping Gu <youpgu71@gmail.com>, Xi Lin <erix025@outlook.com>, Weijie Wang <wangweijie@zju.edu.cn>, Bohan Zhuang <bohan.zhuang@gmail.com>.

720p clip with 81 frames using the Wan2.1–14B (Wan et al., 2025) model, inference on a single NVIDIA H20 GPU can take nearly *two hours*, with attention operations alone accounting for over 80% of the total runtime. This prohibitive cost underscores the urgent need for more efficient attention mechanisms capable of handling long-context video inputs.

To alleviate this computational burden, recent studies exploit the inherent sparsity of the attention map $P = \text{Softmax}(QK^\top / \sqrt{d})$, where most elements become negligible after softmax normalization (Deng et al., 2024). Sparse attention methods leverage this property by computing only the informative regions of P , thereby reducing complexity without sacrificing much accuracy (Zhang et al., 2025a;c; Yang et al., 2025; Xi et al., 2025; Xu et al., 2025). In practice, the dominant paradigm combines a *mask generation strategy* with a *Block Sparse Attention (BSA)* kernel that performs attention only on selected blocks for hardware efficiency (Dao et al., 2022; Dao, 2024; Shah et al., 2024). This paradigm has proven highly effective in reducing attention cost while preserving generation quality, and a series of methods following this paradigm such as XAttention (Xu et al., 2025) and SpargeAttention (Zhang et al., 2025a), have demonstrated excellent efficiency–quality trade-offs in large-scale video generation and understanding tasks.

However, existing block sparse attention mechanisms suffer from a fundamental limitation. At high sparsity levels, their hard binary keep/drop masks severely restrict the key–value region visible to each query, forcing the model to discard many informative areas and leading to significant information loss and performance degradation. Recent work such as Sparse VideoGen(SVG) (Xi et al., 2025; Yang et al., 2025), Sliding Tile Attention (Zhang et al., 2025c) introduce token permutation to concentrate more important key–value tokens within the limited visible block region of each query, thereby partially alleviating this issue. But this strategy conflicts with the design of the causal attention mask: after permutation, the causal mask becomes highly unstructured, making the algorithm difficult to implement efficiently. Moreover, the additional cost of reordering partially offsets the computational gains it aims to achieve.

To this end, we propose *Pyramid Sparse Attention (PSA)*, a module that preserves rich contextual information under low compute budget while remaining compatible with both causal and bi-directional attention masking.

The design of PSA is motivated by an empirical observation that adjacent key tokens in visual sequences exhibit strong local similarity (Figure 2), suggesting that nearby keys can be aggregated by average pooling with minimal information loss. Moreover, a larger pooling size indicates a larger degree of information abstraction. Building on this insight, PSA replaces the binary keep/drop rule in BSA with *multi-level pooled KV representations*, where important KV

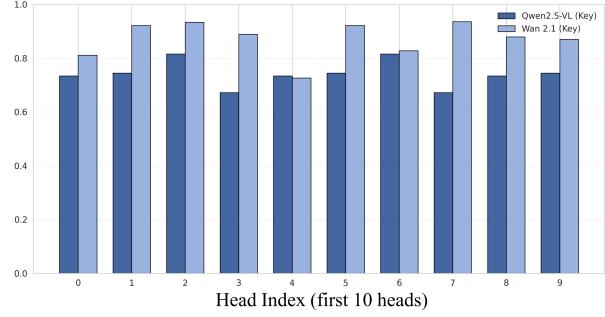


Figure 2. Adjacent Key Token Cosine Similarity. High cosine similarity between key tokens (Qwen2.5-VL, Wan2.1-1.3B) motivates hierarchical pooling: nearby visual tokens are highly similar.

blocks are assigned to lower (finer) pooling levels and less important blocks to higher (coarser) levels, and the least important blocks are entirely skipped to avoid redundant computation (see Figure 3), resulting in a smoother computation allocation under the same compute budget, as shown in Figure 1.

An intuitive analogy can be drawn to Feature Pyramid Networks (FPNs) (Lin et al., 2017) in dense prediction tasks, which assign objects of different scales to distinct feature levels. From a quantization perspective, PSA generalizes BSA’s 1-bit binary mask into a multi-bit, fixed-point scheme. Here, each non-zero element indicates a specific pooling level for the KV block, while a zero value skips the block entirely, enabling finer-grained compute budget allocation.

Moreover, PSA allows each query block to allocate computation adaptively by estimating the importance of each query-key block pair based on attention scores and generating a multi-level attention mask accordingly.

As a result, as shown in Figure 1, under *the same compute budget*, each query in PSA attends to about 70% of the KV blocks on average, substantially expanding its receptive field, while BSA retains only 20% of the blocks and rigidly prunes context. Consequently, PSA produces attention outputs that are much closer to full attention, achieving a superior balance between accuracy and efficiency.

We empirically observe that PSA exhibits broad applicability across both video understanding and video generation tasks. On Video-MME (Fu et al., 2025a) with Qwen2.5-VL (Qwen et al., 2025), PSA matches the full-attention accuracy while achieving the minimal compute budget that preserves quality (approximately 35% full attention FLOPs) among all sparse-attention baselines. For video generation, PSA consistently outperforms prior block-sparse mechanisms in the training-free setting and further improves VBench (Huang et al., 2024) scores when combined with the distillation framework TDM (Luo et al., 2025) on CogVideoX-5B (Hong et al., 2023), even when limited to

only 15% of the full-attention compute budget. These results highlight PSA’s strong compatibility and plug-and-play efficiency across diverse architectures and tasks.

Our contributions are summarized as follows: (1) *Multi-level retention beyond binary masks*: PSA constructs a pyramid of pooled KV blocks that expands each query’s receptive field without increasing FLOPs, thereby mitigating performance degradation under low compute budget. (2) *Hardware-friendly implementation*: We adopt an efficient merge–split kernel design that decouples the logical block size from the hardware tile size, making the processing of dynamically varying KV block sizes introduced by multi-level pooling more efficient. This design maintains efficient GPU utilization, supports backpropagation, and is fully compatible with FlashAttention (Dao, 2024; Dao et al., 2022). (3) *Versatile applicability*: Owing to our design that does not rely on token reordering, PSA can be seamlessly applied to both video understanding and generation models, consistently outperforming all Block-Sparse Attention based methods under low compute budget conditions.

2. Related Work

The quadratic computational and memory cost of standard attention presents a significant bottleneck for processing long sequences, particularly in tasks like video generation (Kong et al., 2025; Wan et al., 2025; Tan et al., 2025; Polyak et al., 2025; Hong et al., 2023) and understanding (Qwen et al., 2025; Zhang et al., 2023a; Madan et al., 2024; Xu et al., 2021; Yan et al., 2021). To overcome this limitation, sparse attention mechanisms have been developed, which apply a mask to the attention matrix to reduce computations. These methods are broadly classified as static or dynamic.

Static sparse attention. Static sparsity methods employ predefined, input-agnostic attention patterns (Zhang et al., 2025c; Hassani et al., 2023; Child et al., 2019; Li et al., 2025; Xiao et al., 2024b; Zhang et al., 2023b; Xiao et al., 2025; 2024a; Chen et al., 2025). These include established patterns such as sliding-window attention (Zhang et al., 2025c; Hassani et al., 2023; Zhang et al., 2023b; Xiao et al., 2024a; 2025), attention sink patterns (Zhu et al., 2025; Fu et al., 2025b; Xiao et al., 2024b), and spatiotemporal energy decay patterns (Li et al., 2025). While computationally efficient, the primary drawback of these methods is their inherent rigidity. Because the patterns are fixed and input-agnostic, they risk missing critical long-range dependencies, which can lead to sub-optimal performance and potentially unstable generation quality.

Dynamic sparse attention. To address the limitations of static patterns, dynamic sparsity methods were introduced. These methods generate an *input-sensitive* mask M during

the forward pass, for instance, by thresholding attention scores (Zhang et al., 2025b; Gu et al., 2025; Xu et al., 2025; Yang et al., 2025; Xi et al., 2025; Lu et al., 2025; Yuan et al., 2025; Song et al., 2025). However, while these content-aware, element-wise adaptive masks offer greater flexibility, they introduce a new challenge: hardware inefficiency. The unstructured, sparse matrix operations resulting from these fine-grained masks lead to poor memory access patterns and low hardware utilization, particularly on parallel processors like GPUs.

Block sparse attention. The challenge of hardware utilization motivated the development of Block Sparse Attention (BSA) (Dao, 2024; Dao et al., 2022; Xu et al., 2025; Gu et al., 2025; Guo et al., 2024). This approach introduces a hardware-aware sparse structure by partitioning the Q , K , V and attention matrices into coarse-grained blocks. A binary block mask M then dictates whether an entire block’s computation is performed or skipped. By operating at this block level, BSA preserves the dense matrix operations within each block, which is essential for achieving high throughput on modern GPUs. However, BSA still relies on a rigid, binary decision at the block level, which can lead to significant information loss under high sparsity. In contrast, our work introduces a multi-level pooled KV pyramid, allowing each query to access a substantially larger receptive field under the same computational budget and critically mitigating the information loss endemic to high-sparsity binary masking. Moreover, our design does not rely on token reordering, ensuring versatile and seamless applicability across both video understanding and generation models while maintaining high GPU throughput via an efficient kernel.

3. Methodology

We propose **Pyramid Sparse Attention (PSA)**, a hierarchical sparse attention framework that allocates computation adaptively across multi-level pooled key-value (KV) representations. PSA consists of three components: (1) Pyramid KV Blocks that capture coarse-to-fine context, (2) a Multi-Level Mask Generator that predicts hierarchical sparsity, and (3) Adaptive Pyramid Attention that fetches KV blocks and computes attention accordingly (see Figure 3).

3.1. Problem Definition and Notation

We consider the standard attention formulation with query, key, and value sequences $Q \in \mathbb{R}^{N \times d}$, $K, V \in \mathbb{R}^{N \times d}$, where N is the sequence length and d is the hidden dimension. The full attention is computed as

$$\text{Attn}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V, \quad (1)$$

whose quadratic complexity $O(N^2)$ becomes the bottleneck for long sequences, for example in video understanding or

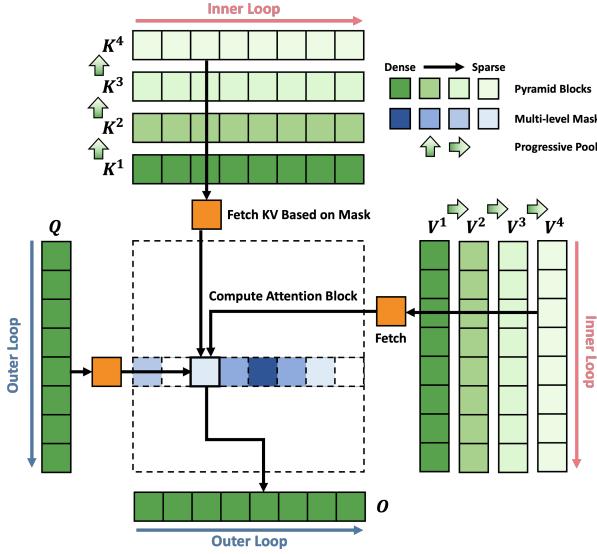


Figure 3. Overview of the Pyramid Sparse Attention (PSA) framework. PSA adaptively allocates attention computation across hierarchical KV representations (green; lighter shades denote coarser levels). The multi-level mask (blue) determines which KV level each query block attends to. As illustrated, the current attention block assigned to level 4 uses the coarsest KV representation K_j^4 and V_j^4 .

generation.

To reduce computation and scale attention to long sequences, block sparse attention is commonly adopted to restrict attention computation to a subset of token blocks. The query and KV sequences are divided into non-overlapping blocks of size b_q and b_k , resulting in $n_q = N/b_q$ query blocks and $n_k = N/b_k$ KV blocks. The attention is then computed block-wise between query blocks $\{Q_i\}_{i=1}^{n_q}$ and KV blocks $\{(K_j, V_j)\}_{j=1}^{n_k}$.

A binary attention mask $M \in \{0, 1\}^{n_q \times n_k}$ is typically used in block sparse attention, where $M_{ij} = 1$ indicates that query block Q_i attends to KV block (K_j, V_j) . However, such binary masking enforces a hard keep-or-drop decision, which not only causes information loss from discarded regions but also limits how computation can be flexibly allocated across blocks of varying importance.

To address this limitation, we introduce a multi-level representation of KV blocks and a multi-level mask $M \in \{0, 1, \dots, H\}^{n_q \times n_k}$, allowing a hierarchical and adaptive sparsity control. This serves as the foundation of our proposed PSA, which dynamically allocates compute budget across coarse-to-fine levels. We detail the key design components in the following sections.

3.2. Pyramid KV Blocks

To construct multi-level KV representations, we build a hierarchical pyramid of H levels by progressively pooling

along the sequence dimension:

$$K_i^1 = K_i, \quad K_i^{h+1} = \text{MeanPool}(K_i^h, 2, 2), \quad (2)$$

$$V_i^1 = V_i, \quad V_i^{h+1} = \text{MeanPool}(V_i^h, 2, 2), \quad (3)$$

where $\text{MeanPool}(x, k, s)$ denotes 1D mean pooling with kernel size k and stride s along the sequence dimension.

Through this process, we obtain sets of pyramid KV blocks $\{K_i^1, K_i^2, \dots, K_i^H\}$ and $\{V_i^1, V_i^2, \dots, V_i^H\}$, where the h -th level block represents a token group of size $\frac{b_k}{2^{h-1}}$. As h increases, the representation becomes increasingly coarser, summarizing broader contextual information within each block. This hierarchical pooling reduces the effective KV length by a factor of 2^{h-1} at pyramid level h , enabling the model to access coarse-to-fine contextual representations and dynamically balance accuracy and efficiency under limited compute budget.

3.3. Multi-Level Mask Generator

Given the multi-level KV hierarchy, we estimate the importance of each query–key block pair and generate a multi-level mask to guide adaptive attention computation.

3.3.1. BLOCK IMPORTANCE ESTIMATION

In this step, we compute a lightweight importance score S_{ij} for each query–key block pair (Q_i, K_j) , which reflects how crucial it is to preserve fine-grained attention information within that region. To leverage domain-specific characteristics, we employ different importance estimators for video generation and understanding tasks.

For **video generation**, we employ a sampling-based strategy to approximate block importance efficiently. Specifically, a small subset of tokens is randomly sampled from both the query and key blocks, denoted as $\tilde{Q}_i \in \mathbb{R}^{s_q \times d}$ and $\tilde{K}_j \in \mathbb{R}^{s_k \times d}$, where $s_q \ll b_q$ and $s_k \ll b_k$. The importance score is then estimated as the maximum attention score between the sampled tokens:

$$S_{ij} = \max \left(\text{Softmax} \left(\frac{\tilde{Q}_i \tilde{K}_j^\top}{\sqrt{d}} \right) \right). \quad (4)$$

To improve locality and stabilize both the sampling estimation and the pyramid pooling, we apply a Hilbert curve permutation (Zhang et al., 2025a) to the base sequences before computation.

For **video understanding**, we adopt the antidiagonal scoring (Xu et al., 2025), and introduce intra-block similarity verification to address the relatively low token similarity in video understanding (Figure 2). This verification compares the cosine similarity of adjacent tokens against level-specific thresholds, which directly sets a maximum acceptable pooling level for the block.

Algorithm 1 Computation of PSA

Require: Query blocks $\{Q_i\}_{i=1}^{n_q}$, pyramid KV blocks $\{K_j^h, V_j^h\}_{j=1, h=1}^{n_k, H}$, mask M , scale factor $1/\sqrt{d}$

Ensure: Output blocks $\{O_i\}_{i=1}^{n_q}$

- 1: **for** each query block Q_i **do**
- 2: Initialize $o_i \leftarrow 0, m_i \leftarrow -\infty, l_i \leftarrow 0$
- 3: **for** each key block K_j **do**
- 4: $h \leftarrow M_{ij}$
- 5: **if** $h = 0$ **then**
- 6: **continue** {Skip pruned block}
- 7: **end if**
- 8: $S_{ij} \leftarrow Q_i K_j^{h\top} / \sqrt{d} + (h-1) \ln 2$
- 9: $m_{ij} \leftarrow \max(\text{rowmax}(S_{ij}), m_i)$
- 10: $P_{ij} \leftarrow \exp(S_{ij} - m_{ij})$
- 11: $l_{ij} \leftarrow l_i \exp(m_i - m_{ij}) + \text{rowsum}(P_{ij})$
- 12: $o_i \leftarrow o_i \exp(m_i - m_{ij}) + P_{ij} V_j^h$
- 13: $m_i \leftarrow m_{ij}, l_i \leftarrow l_{ij}$
- 14: **end for**
- 15: $O_i \leftarrow o_i / l_i$
- 16: **end for**
- 17: **return** $\{O_i\}_{i=1}^{n_q}$

These variants demonstrate that PSA is agnostic to the specific importance estimator and readily adapts to various architectures and tasks.

3.3.2. MULTI-LEVEL MASK ASSIGNMENT

Based on the estimated importance scores S , we generate a multi-level mask M to guide adaptive attention computation, as detailed in Algorithm 2.

We introduce a *multi-level mask* to assign pyramid levels dynamically. Each entry $M_{ij} \in \{0, 1, 2, \dots, H\}$ specifies which level of pyramid KV block (K_j, V_j) should be fetched when computing attention for the query block Q_i :

$$M_{ij} = h > 0 \Rightarrow \text{use } K_j^h, V_j^h, \quad M_{ij} = 0 \Rightarrow \text{skip.} \quad (5)$$

A larger h indicates lower block importance and coarser KV representations, while $M_{ij} = 0$ corresponds to skipping the block. This formulation allows a gradual degradation of precision instead of a binary drop, effectively mitigating abrupt information loss at high sparsity.

We leverage a threshold-based masking strategy to flexibly control the sparsity among query blocks. Specifically, we first normalize the importance scores S row-wise, and then compute the cumulative scores \hat{S}_{ij} by summing the descending sorted importance scores for each query block.

To translate these cumulative scores into multi-level sparsity, we define a set of thresholds as hyper-parameters that determine the pyramid level assignment. For a pyramid with H

Algorithm 2 Multi-Level Mask Assignment

Require: Importance map $S \in \mathbb{R}^{n_q \times n_k}$, thresholds $T = \{\tau_1, \dots, \tau_H\}$

Ensure: Multi-level mask $M \in \{0, \dots, H\}^{n_q \times n_k}$

- 1: $M \leftarrow \text{zeros_like}(S)$
- 2: **for** $i = 1$ **to** n_q **do**
- 3: $E_i \leftarrow S_i / \sum_j S_{ij}$ {Normalize row}
- 4: $(E'_i, \pi_i) \leftarrow \text{sort_desc}(E_i)$
- 5: $\hat{E}_i \leftarrow \text{cumsum}(E'_i)$
- 6: **for** $j = 1$ **to** n_k **do**
- 7: $j' \leftarrow \pi_i(j)$ {Map to sorted index}
- 8: **if** $\hat{E}_{ij'} > \tau_H$ **then**
- 9: $M_{ij} \leftarrow 0$
- 10: **else**
- 11: $M_{ij} \leftarrow \min\{t \mid \hat{E}_{ij'} \leq \tau_t\}$
- 12: **end if**
- 13: **end for**
- 14: **end for**
- 15: **return** M

levels, the thresholds are defined as:

$$T = \tau_1, \tau_2, \dots, \tau_H, \quad 0 \leq \tau_1 \leq \tau_2 \leq \dots \leq \tau_H \leq 1. \quad (6)$$

These thresholds specify the score budgets allocated to each level, enabling fine-grained sparsity control. The pyramid level for each query-key block pair is assigned as:

$$M_{ij} = \begin{cases} \min\{t \mid \hat{S}_{ij} \leq \tau_t\}, & \text{if } \hat{S}_{ij} \leq \tau_H \\ 0, & \text{if } \hat{S}_{ij} > \tau_H. \end{cases} \quad (7)$$

Unlike quantile-based strategies with fixed per-level quotas, our threshold-based strategy dynamically adjusts sparsity according to the cumulative importance score distribution of each query. This adaptive mechanism allows flexible computation allocation and leads to more accurate attention estimation (see Section 4.5 for analysis).

3.4. Adaptive Pyramid Attention

With the pyramid KV representations and the multi-level mask constructed, we now compute the attention in a block-wise manner. For each block, we fetch the corresponding key/value block based on the assigned pyramid level in the mask according to Equation (5) and compute attention accordingly.

However, KV blocks in level h have reduced sequence lengths due to pooling, meaning each token in K_j^h represents an aggregated context of 2^{h-1} original tokens. To maintain consistent probability distribution in attention weights after softmax, we introduce a scaling factor of 2^{h-1} to the attention scores. This is efficiently implemented by adding a



Figure 4. Qualitative comparison on Wan2.1-1.3B (Text-to-Video, 720p).

bias term $(h - 1) \ln 2$ to the attention logits before softmax normalization.

Specifically, the attention score between query block Q_i and key block K_j at level h is computed as:

$$A_{ij} = \text{Softmax}\left(\frac{Q_i K_j^{h\top}}{\sqrt{d}} + (h - 1) \ln 2\right). \quad (8)$$

The mechanism is detailed in Algorithm 1.

Complexity analysis. At pooling level h , the effective sequence length of each KV block is reduced by a factor of 2^{h-1} , resulting in an attention cost of $O(b_q b_k / 2^{h-1})$ for that attention block.

Aggregating across all levels based on the mask distribution yields an overall expected complexity of $O(\bar{\rho} N^2)$, where $\bar{\rho}$ is the effective sparsity ratio induced by the multi-level mask:

$$\bar{\rho} = \frac{1}{n_q n_k} \sum_{i=1}^{n_q} \sum_{j=1}^{n_k} \rho_{ij}, \text{ with } \rho_{ij} = \begin{cases} \frac{1}{2^{M_{ij}-1}}, & M_{ij} > 0 \\ 0, & M_{ij} = 0 \end{cases} \quad (9)$$

Compared with standard block-sparse-based attention mechanisms, PSA distributes the same compute budget more efficiently—assigning finer-grained attention to informative regions and coarser attention to redundant ones. This adaptive allocation allows the model to retain more critical details under the same computational cost, improving representational fidelity without increasing total FLOPs.

3.5. Hardware-Optimized Kernel Design

To ensure that PSA can be deployed efficiently on modern accelerators, we complement the algorithmic design with a hardware-aware implementation. The key challenge is that existing implementations often couple the logical block size with the hardware tile size, although these two hyperparameters serve fundamentally different purposes. The block size captures the logical grouping of tokens while the tile size must be selected to maximize hardware throughput. This mismatch becomes especially pronounced in PSA because KV block pooling produces heterogeneous and often small blocks at coarser pyramid levels. Using these various block sizes directly as the execution tile leads to low compute utilization or warp divergence when the tile configuration changes dynamically.

To resolve this issue, we adopt a **decoupled block-tile design** that separates logical sequence blocks from the execution tiles used by the kernel. Building on a modified FlashAttention kernel, blocks are flexibly split or merged to match a hardware-optimized tile size. This approach allows the block size to follow the attention pattern while the tile size is tuned independently for accelerator efficiency. The design integrates seamlessly with kernel fusion, optimized memory access, and fine-grained parallelization.

The decoupled design offers broad compatibility with attention mechanisms for blocked sequences, enabling them to handle heterogeneous block structures while maintaining hardware efficiency. In practice, our kernel preserves high tensor-core utilization even for small pooled blocks, avoids divergence across pooling levels, and achieves up to a **10×** speedup over a naive PSA implementation on NVIDIA H200.

Table 1. Quantitative comparison on Wan-series models in training-free videoegen experiments. We report similarity metrics (PSNR, SSIM, LPIPS) and perceptual quality measures (Aesthetic Quality (Aes.), Background Consistency (Bkg.), and Imaging Quality (Img.)) from VBench (Huang et al., 2024). Latency represents the average generation time per video. For clarity, the **best** result among all sparse methods under each metric is **bolded**, and the **second-best** is underlined.

Model	Method	PSNR↑	SSIM↑	LPIPS↓	Aes.↑	Bkg.↑	Img.↑	Sparsity	Latency(s)
Wan 2.1 1.3B	Full	–	–	–	0.6489	0.9645	0.6557	–	327
	SVG2	25.21	0.801	0.126	0.6185	0.9548	0.5545	0.91	187
	SVG	17.57	0.567	0.399	0.5039	0.9444	0.5974	0.85	165
	Sparge	22.83	0.736	0.177	0.6232	0.9476	0.6409	0.90	165
	STA	20.56	0.677	0.197	<u>0.6521</u>	0.9419	<u>0.6501</u>	0.83	162
	PSA(Ours)	24.36	0.788	0.121	0.6686	0.9612	0.6607	0.91	176
Wan 2.2 5B (1280 × 704, 121 frames)	Full	–	–	–	0.6598	0.9564	0.6547	–	168
	SVG2	24.25	0.818	0.092	0.6495	0.9518	<u>0.6025</u>	0.90	149
	SVG	18.89	0.645	0.266	0.5539	0.9386	0.5877	0.86	122
	Sparge	19.53	0.660	0.229	0.5482	0.9289	0.5650	0.89	124
	PSA(Ours)	23.03	0.794	<u>0.096</u>	0.6588	0.9569	0.6438	0.89	131
Wan 2.1 14B	Full	–	–	–	0.6918	0.9639	0.6247	–	1548
	SVG2	24.79	0.807	0.085	0.6614	0.9439	0.5555	0.87	913
	SVG	19.84	0.649	0.300	0.5337	0.9501	0.5479	0.85	830
	Sparge	22.19	0.737	0.182	0.6083	0.8779	0.5977	0.88	855
	STA	20.83	0.694	0.185	0.6544	0.9399	0.6489	0.83	815
	PSA(Ours)	23.83	0.768	<u>0.105</u>	0.6776	0.9261	0.6400	0.88	887

4. Experiments

4.1. Experimental Settings

Models. We evaluate PSA on open-source video generation models and video understanding models: Wan2.1-{1.3B, 14B} (Wan et al., 2025), Wan2.2-5B (Wan et al., 2025), CogVideoX-5B (Hong et al., 2023), and Qwen2.5-VL-7B (Qwen et al., 2025).

Baselines. We compare our method, Pyramid Sparse Attention (PSA), against several representative sparse attention baselines, including Sparse VideoGen (SVG) (Xi et al., 2025), Sparse VideoGen2 (SVG2) (Yang et al., 2025), Sliding-Tile Attention (STA) (Zhang et al., 2025c), SpargeAttention (Sparge) (Zhang et al., 2025a), and XAttention (Xu et al., 2025). For brevity, we use abbreviated names (e.g., SVG2, STA, PSA) throughout the remainder of this section.

Implementation details. All experiments are conducted on NVIDIA H200 GPUs. Unless stated, videos are generated at 1280×768 / 69 frames. All sparse baselines are evaluated using their official implementations without any additional optimization tricks. In video understanding experiments, PSA reuses the same block-importance estimation based on antidiagonal scoring and incorporates an additional similarity-based constraint, as detailed in Section 3.3.1 . All sparse attention mechanisms are applied only during the prefill stage.

On sparsity accounting. PSA allocates multi-level compute to KV blocks rather than using binary keep/drop strategy; its reported sparsity thus denotes the sparsity of BSA-

based method that uses the same compute budget.

Metrics and dataset. For the training-free video generation experiments, we evaluate the similarity between the generated videos and their full-attention counterparts using Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al., 2018). Beyond these similarity measures, we further adopt three perceptual dimensions from the VBench Score (Huang et al., 2024)—widely used in recent video generation benchmarks to assess visual quality: Aesthetic Quality (Aes.), Background Consistency (Bkg.), and Imaging Quality (Img.).

For the distillation experiments on CogvideoX-5B (Hong et al., 2023), we primarily use the VBench Score to measure the perceptual quality of videos generated by distilled models. Our distillation process is guided by a dataset of 10,000 text prompts, sampled from the JourneyDB benchmark (Sun et al., 2023). To enhance prompt quality and diversity, each sample is refined using the Qwen2.5-3B-Instruct model (Qwen et al., 2025).

For video understanding, we adopt the Video-MME (1fps) dataset (Fu et al., 2025a) to evaluate the performance of Qwen2.5-VL-7B in video understanding scenarios.

4.2. Training-Free Video Generation

A qualitative comparison of sparse attention mechanisms is shown in Figure 4. All methods use the same text-to-video scene (a couple walking under a red umbrella in the rain) under similar sparsity. STA (Zhang et al., 2025c) exhibits

Table 2. Combining PSA with TDM on CogVideoX-5B.

Method	Sparsity	Sampling Steps	VBench Score
FullAttn	—	50	0.819
Distill-only	—	4	0.818
Ours	0.85	4	0.826

Table 3. Comparison of attention mechanisms on Video-MME.

Method	Short	Medium	Long	Overall	Sparsity
Full Attention	0.752	0.663	0.537	0.651	—
XAttention	0.748	0.661	0.544	0.651	0.58
SpurgeAttention	0.749	0.663	0.539	0.650	0.37
PSA(Ours)	0.748	0.673	0.542	0.654	0.65

structure jumps and identity swaps across frames; SVG (Xi et al., 2025) collapses at high sparsity with random color blocks; SVG2 (Yang et al., 2025) is more stable but overly smooth with distorted backgrounds; Spurge (Zhang et al., 2025a) preserves layout yet shows local color distortions; Full Attention remains high-fidelity but costly. In contrast, PSA at 0.91 sparsity maintains sharp details and temporal coherence (stable contours, saturated umbrella, natural reflections), approaching the dense baseline’s visual quality with far less computation.

Table 1 further shows that under comparable high sparsity, PSA consistently surpasses SVG, STA, and Spurge on all similarity metrics (PSNR/SSIM/LPIPS) and on most perceptual axes (Aes./Bkg./Img.) across model sizes. Against the strongest training-free baseline SVG2, PSA delivers *clearly better perceptual quality* at high sparsity—higher Aesthetic and Imaging scores on *all* model sizes —while keeping similarity metrics within small margins. Together with the visuals in Figure 4, these results indicate that PSA is the *most quality-preserving* sparse attention mechanism at high sparsity among all compared methods. Additional visual comparisons are provided in the appendix.

4.3. Distillation

To explore the generality of PSA and achieve maximum inference acceleration, we combined our method with the *data-free* distillation technique TDM (Luo et al., 2025; Gu et al., 2025). We integrated PSA into the student model during the distillation training phase. As shown in Table 2, this simple combination achieves a $30.2\times$ denoising time speedup over the original 50-step model, without any loss in generation quality.

Our combined approach, using 85% sparsity, achieves the highest VBench score (0.826). This surpasses both the 4-step distilled baseline and even the original 50-step model. This result demonstrates that PSA is a highly compatible plug-and-play module that can be compounded with other methods like distillation to maximize inference efficiency.

Table 4. Comparison of multi-level vs. binary masking on Wan2.1-1.3B (480p, training-free). The first 25% of sampling steps do not use sparse attention. $T = \{\tau_1, \dots, \tau_H\}$ are the cumulative thresholds for level assignment.

Method	Sparsity↑	PSNR↑	SSIM↑	LPIPS↓
Multi-level mask	0.79	23.35	0.856	0.116
Binary mask	0.75	23.11	0.851	0.122

Table 5. Ablation of reordering and mask-generation strategy on Wan2.1-1.3B (480p).

Mode	Reorder	Sparsity↑	PSNR↑	SSIM↑	LPIPS↓
Threshold-based	Hilbert	0.84	23.78	0.811	0.085
Quantile-based	Hilbert	0.85	22.54	0.775	0.118
Quantile-based	None	0.85	21.81	0.707	0.201

4.4. Training-Free Video Understanding

On Qwen2.5-VL-7B, PSA delivers the best overall performance on Video-MME (Fu et al., 2025a) under **the highest sparsity level (0.65)**. As shown in Table 3, it matches or exceeds the full baseline across most categories—notably improving the *Medium* and *Long-Video* scores—while remaining competitive on *Short Video*. These results show that PSA preserves (and in some cases improves) video-understanding quality at substantially higher sparsity, demonstrating the strongest quality-retaining sparsity among all compared methods in the training-free setting.

4.5. Ablation Study

Effectiveness of multi-level mask vs. binary mask. In terms of configuration, the multi-level mask group is set to $T = \{0.70, 0.80, 0.90, 0.90\}$, while binary mask uses level ratios $T = \{0.85, 0.85, 0.85, 0.85\}$, which means that only dense KV blocks are kept and no pooled representation is used.

On Wan2.1-1.3B (480p, training-free), we compare our multi-level masking with a conventional 0/1 binary mask. The first 25% of sampling steps do not use sparse attention. As shown in Table 4, PSA yields improved PSNR and SSIM, and reduced LPIPS, despite operating under a higher sparsity setting than the baselines.

Reordering & mask strategy (Videogen). We evaluate different reordering and mask-generation strategies on Wan2.1-1.3B (480p, training-free setting) and the first 25% of sampling steps do not use sparse attention.

We compare two multi-level masking rules (threshold-based vs. quantile-based) and the role of token reordering. The quantile-based rule processes each query row by sorting block importance $S_{i,:}$ and assigning pooling degrees by fixed percentile cut points $0 < a < b < c \leq d \leq 1$ (e.g., $[0, a] \rightarrow 1$, $[a, b] \rightarrow 2$, $[b, c] \rightarrow 3$, $[c, d] \rightarrow 4$, $(d, 1] \rightarrow$

0). For reordering, we follow SpARGEAttention and apply a HilbertCurve permutation to Q, K, V (Zhang et al., 2025a). Hilbert denotes a Hilbert-curve-based permutation, while None indicates no reordering.

As shown in Table 5, under near-matched sparsity, Threshold-based + reordering consistently outperforms quantile-based + reordering in reconstruction quality. Removing the reordering step from the quantile-based variant further degrades results. As observed in SpARGEAttention (Zhang et al., 2025a), this reordering operation enhances the similarity between adjacent tokens. When incorporated into PSA, this property naturally complements the hierarchical pooling mechanism, as higher local token similarity leads to smoother multi-level aggregation and more stable reconstruction under high sparsity. In contrast to the quantile-based rule, the threshold-based strategy dynamically adjusts the proportion of each mask based on the input distribution, allowing PSA to better adapt to diverse sparsity patterns and preserve visual fidelity.

Additional ablations on the similarity constraint and variable block sizes, which also influence the performance of PSA, are presented in the supplementary material

5. Conclusion

We propose Pyramid Sparse Attention (PSA), a versatile sparse attention mechanism validated across both video understanding and video generation tasks. Our method introduces multi-level sparsity beyond binary keep/skip choices, where higher sparsity levels use larger pooling degrees of KV representations; in this way, unlike block-sparse attention that fully drops low-score blocks, we preserve substantially more information under the same compute budget. As a result, PSA demonstrates strong generalization and consistently outperforms all existing Block Sparse Attention based methods under low compute budget, highlighting its robustness across diverse model architectures and application scenarios.

References

- Chen, P., Zeng, X., Zhao, M., Ye, P., Shen, M., Cheng, W., Yu, G., and Chen, T. Sparse-vdit: Unleashing the power of sparse attention to accelerate video diffusion transformers, 2025. URL <https://arxiv.org/abs/2506.03065>.
- Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers, 2019. URL <https://arxiv.org/abs/1904.10509>.
- Dao, T. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.
- Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.
- Deng, Y., Song, Z., and Yang, C. Attention is naturally sparse with gaussian distributed input. *CoRR*, 2024.
- Fu, C., Dai, Y., Luo, Y., Li, L., Ren, S., Zhang, R., Wang, Z., Zhou, C., Shen, Y., Zhang, M., et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 24108–24118, 2025a.
- Fu, T., Huang, H., Ning, X., Zhang, G., Chen, B., Wu, T., Wang, H., Huang, Z., Li, S., Yan, S., et al. Moa: Mixture of sparse attention for automatic large language model compression. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025b.
- Gu, Y., Li, X., Hu, Y., Chen, M., and Zhuang, B. Blade: Block-sparse attention meets step distillation for efficient video generation, 2025. URL <https://arxiv.org/abs/2508.10774>.
- Guo, J., Tang, H., Yang, S., Zhang, Z., Liu, Z., and Han, S. Block Sparse Attention. <https://github.com/mit-han-lab/Block-Sparse-Attention>, 2024.
- Hassani, A., Walton, S., Li, J., Li, S., and Shi, H. Neighborhood attention transformer. In *CVPR*, pp. 6185–6194, June 2023.
- Hong, W., Ding, M., Zheng, W., Liu, X., and Tang, J. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. In *ICLR*, 2023.
- Huang, Z., He, Y., Yu, J., Zhang, F., Si, C., Jiang, Y., Zhang, Y., Wu, T., Jin, Q., Chanpaisit, N., et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21807–21818, 2024.
- 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. Hunyuanyvideo: A systematic framework for large video generative models, 2025. URL <https://arxiv.org/abs/2412.03603>.

- Li, X., Li, M., Cai, T., Xi, H., Yang, S., Lin, Y., Zhang, L., Yang, S., Hu, J., Peng, K., Agrawala, M., Stoica, I., Keutzer, K., and Han, S. Radial attention: $o(n \log n)$ sparse attention with energy decay for long video generation, 2025. URL <https://arxiv.org/abs/2506.19852>.
- Lin, B., Ye, Y., Zhu, B., Cui, J., Ning, M., Jin, P., and Yuan, L. Video-llava: Learning unified visual representation by alignment before projection. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 5971–5984, 2024.
- Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.
- Lu, E., Jiang, Z., Liu, J., Du, Y., Jiang, T., Hong, C., Liu, S., He, W., Yuan, E., Wang, Y., Huang, Z., Yuan, H., Xu, S., Xu, X., Lai, G., Chen, Y., Zheng, H., Yan, J., Su, J., Wu, Y., Zhang, N. Y., Yang, Z., Zhou, X., Zhang, M., and Qiu, J. Moba: Mixture of block attention for long-context llms, 2025. URL <https://arxiv.org/abs/2502.13189>.
- Luo, Y., Hu, T., Sun, J., Cai, Y., and Tang, J. Learning few-step diffusion models by trajectory distribution matching. In *ICCV*, 2025.
- Madan, N., Moegelmose, A., Modi, R., Rawat, Y. S., and Moeslund, T. B. Foundation models for video understanding: A survey, 2024. URL <https://arxiv.org/abs/2405.03770>.
- Peebles, W. and Xie, S. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.
- Polyak, A., Zohar, A., Brown, A., Tjandra, A., Sinha, A., Lee, A., Vyas, A., Shi, B., Ma, C.-Y., Chuang, C.-Y., Yan, D., Choudhary, D., Wang, D., Sethi, G., Pang, G., Ma, H., Misra, I., Hou, J., Wang, J., Jagadeesh, K., Li, K., Zhang, L., Singh, M., Williamson, M., Le, M., Yu, M., Singh, M. K., Zhang, P., Vajda, P., Duval, Q., Girdhar, R., Sumbaly, R., Rambhatla, S. S., Tsai, S., Azadi, S., Datta, S., Chen, S., Bell, S., Ramaswamy, S., Sheynin, S., Bhattacharya, S., Motwani, S., Xu, T., Li, T., Hou, T., Hsu, W.-N., Yin, X., Dai, X., Taigman, Y., Luo, Y., Liu, Y.-C., Wu, Y.-C., Zhao, Y., Kirstain, Y., He, Z., He, Z., Pumarola, A., Thabet, A., Sanakoyeu, A., Mallya, A., Guo, B., Araya, B., Kerr, B., Wood, C., Liu, C., Peng, C., Vengertsev, D., Schonfeld, E., Blanchard, E., Juefei-Xu, F., Nord, F., Liang, J., Hoffman, J., Kohler, J., Fire, K., Sivakumar, K., Chen, L., Yu, L., Gao, L., Georgopoulos, M., Moritz, R., Sampson, S. K., Li, S., Parmeggiani, S., Fine, S., Fowler, T., Petrovic, V., and Du, Y. Movie gen: A cast of media foundation models, 2025. URL <https://arxiv.org/abs/2410.13720>.
- Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2025. URL <https://arxiv.org/abs/2412.15115>.
- Shah, J., Bikshand, G., Zhang, Y., Thakkar, V., Ramani, P., and Dao, T. Flashattention-3: fast and accurate attention with asynchrony and low-precision. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pp. 68658–68685, 2024.
- Song, E., Chai, W., Yang, S., Armand, E., Shan, X., Xu, H., Xie, J., and Tu, Z. Videonsa: Native sparse attention scales video understanding, 2025. URL <https://arxiv.org/abs/2510.02295>.
- Sun, K., Pan, J., Ge, Y., Li, H., Duan, H., Wu, X., Zhang, R., Zhou, A., Qin, Z., Wang, Y., et al. Journeydb: A benchmark for generative image understanding. *Advances in neural information processing systems*, 36:49659–49678, 2023.
- Tan, X., Chen, Y., Jiang, Y., Chen, X., Yan, K., Duan, N., Zhu, Y., Jiang, D., and Xu, H. Dsv: Exploiting dynamic sparsity to accelerate large-scale video dit training, 2025. URL <https://arxiv.org/abs/2502.07590>.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Wan, T., 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. Wan: Open and advanced large-scale video generative models, 2025. URL <https://arxiv.org/abs/2503.20314>.

- Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., Bai, J., Chen, K., Liu, X., Wang, J., Ge, W., et al. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution. *CoRR*, 2024.
- Xi, H., Yang, S., Zhao, Y., Xu, C., Li, M., Li, X., Lin, Y., Cai, H., Zhang, J., Li, D., Chen, J., Stoica, I., Keutzer, K., and Han, S. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. In *Forty-Second International Conference on Machine Learning*, 2025.
- Xiao, C., Zhang, P., Han, X., Xiao, G., Lin, Y., Zhang, Z., Liu, Z., and Sun, M. Inflm: Training-free long-context extrapolation for llms with an efficient context memory. *Advances in Neural Information Processing Systems*, 37: 119638–119661, 2024a.
- Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. In *ICLR*, 2024b.
- Xiao, G., Tang, J., Zuo, J., Yang, S., Tang, H., Fu, Y., Han, S., et al. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. In *The Thirteenth International Conference on Learning Representations*, 2025.
- Xu, H., Ghosh, G., Huang, P.-Y., Okhonko, D., Aghajanyan, A., Metze, F., Zettlemoyer, L., and Feichtenhofer, C. Videoclip: Contrastive pre-training for zero-shot video-text understanding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6787–6800, 2021.
- Xu, R., Xiao, G., Huang, H., Guo, J., and Han, S. Xattention: Block sparse attention with antidiagonal scoring. In *Forty-Second International Conference on Machine Learning*, 2025.
- Yan, W., Zhang, Y., Abbeel, P., and Srinivas, A. Videogpt: Video generation using vq-vae and transformers, 2021. URL <https://arxiv.org/abs/2104.10157>.
- Yang, S., Xi, H., Zhao, Y., Li, M., Zhang, J., Cai, H., Lin, Y., Li, X., Xu, C., Peng, K., Chen, J., Han, S., Keutzer, K., and Stoica, I. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation. In *NeurIPS*, 2025.
- Yuan, J., Gao, H., Dai, D., Luo, J., Zhao, L., Zhang, Z., Xie, Z., Wei, Y., Wang, L., Xiao, Z., et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, 2025.
- Zhang, H., Li, X., and Bing, L. Video-llama: An instruction-tuned audio-visual language model for video understanding. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023a.
- Zhang, J., Xiang, C., Huang, H., Xi, H., Zhu, J., Chen, J., et al. Spageattention: Accurate and training-free sparse attention accelerating any model inference. In *Forty-second International Conference on Machine Learning*, 2025a.
- Zhang, P., Chen, Y., Huang, H., Lin, W., Liu, Z., Stoica, I., Xing, E. P., and Zhang, H. Faster video diffusion with trainable sparse attention. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.
- Zhang, P., Chen, Y., Su, R., Ding, H., Stoica, I., Liu, Z., and Zhang, H. Fast video generation with sliding tile attention. In *Forty-Second International Conference on Machine Learning*, 2025c.
- Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.
- Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., Wang, Z. A., and Chen, B. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710. Curran Associates, Inc., 2023b. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6ceefaa7b15572587b78ecfcabb2827f8-Paper-Conference.pdf.
- Zhu, Q., Duan, J., Chen, C., Liu, S., Li, X., Feng, G., Lv, X., Chuanfu, X., Lin, D., and Yang, C. Sampleattention: Near-lossless acceleration of long context llm inference with adaptive structured sparse attention. In *Eighth Conference on Machine Learning and Systems*, 2025.

A. Appendix

Due to space limitations in the main paper, we provide additional technical details, ablation studies, and qualitative comparisons in this appendix. The material is organized as follows:

- **Sec. B: Implementation Details of PSA.**

Detailed description of PSA modules, including the cosine similarity-based pooling constraint, block size variants, and importance pooling operators.

- **Sec. C: Additional Ablations.**

Comprehensive ablation experiments on cosine similarity, block size, pooling operators, and multi-level allocation strategies across video understanding and video generation tasks.

- **Sec. D: Additional Visual Comparisons.**

Extended qualitative examples for the training-free video generation evaluation, following the same layout as the main paper.

The following sections provide the full details and results.

B. Implementation Details of PSA

B.1. Cosine Similarity Based Pooling Constraint

On top of the importance driven mask M , PSA optionally incorporates a cosine similarity based pooling constraint. This module is inspired by the cosine similarity constraint design in SpurgeAttention (Zhang et al., 2025a), but we generalize it to our multi-level pooling regime. The intuition is that key blocks whose tokens are internally similar can be safely assigned to coarser pyramid levels, whereas blocks with heterogeneous tokens should remain at finer levels.

For PSA with H levels, we introduce $H - 1$ cosine similarity thresholds

$$T_s = \{\tau_s^{(2)}, \tau_s^{(3)}, \dots, \tau_s^{(H)}\},$$

where each $\tau_s^{(h)}$ specifies the minimum intra-block cosine similarity required for a block to be eligible for level h . Given these thresholds, we compute the per-block maximum admissible level $L \in \{1, \dots, H\}^{n_k}$ using Algorithm 3, which evaluates intra-block similarities at strides corresponding to different pooling sizes. A block may enter level h only if its stride- 2^{h-1} similarity exceeds $\tau_s^{(h)}$. The final mask used by PSA is obtained by combining this constraint with the importance-driven mask M via

$$\tilde{M}_{ij} = \min(M_{ij}, L_j),$$

ensuring that no block is assigned a level coarser than what its similarity permits.

In our experiments, we consider a pyramid with $H = 4$ levels and use different similarity thresholds (τ_2, τ_3, τ_4) depending on the task. For Video-MME, we adopt thresholds of $(0.75, 0.70, 0.70)$, while for the video generation preset *PSA (sim)* in Table 7, we use $(0.70, 0.65, 0.60)$. The *no-sim* variant is obtained by setting all thresholds to -1 , which yields $L_j \equiv H$ and disables the similarity constraint entirely. Conversely, the *1-level* variant sets all thresholds to 1 , enforcing $L_j = 1$ for all blocks and collapsing the multi-level structure into a single fine-grained level.

B.2. Block Size and Pooling Variants

The query and key block sizes determine the granularity at which importance scores and masks are computed, and are conceptually independent of the hardware tile sizes used within the kernel implementation. Owing to the decoupled block-tile design introduced in Section 3.5, PSA can employ moderate block sizes (b_q, b_k) (e.g., $(32, 32)$, $(64, 64)$, $(128, 64)$) while still achieving relatively high tensor-core utilization.

To examine the role of the pooling operator in the importance definition (Section 3.3.1), we additionally evaluate a *mean-pooling* variant that replaces the max operator with an average over the sampled attention scores, while keeping all other components (sampling, permutation, mask generation, and similarity constraint) identical. The corresponding ablation results are shown in Table 8.

Algorithm 3 Cosine Similarity-Based Pooling Level Constraint

```

Input: key blocks  $\{K_j\}_{j=1}^{n_k}$ , thresholds  $\tau_s^{(2)}, \dots, \tau_s^{(H)}$ 
Output: maximum admissible levels  $L \in \{1, \dots, H\}^{n_k}$ 
Initialize  $L_j \leftarrow 1$  for all  $j$ 
for  $j = 1$  to  $n_k$  do
    Interpret  $K_j$  as a tensor in  $\mathbb{R}^{B \times n_{\text{head}} \times b_k \times d}$ 
    for  $h = 2$  to  $H$  do
        Compute  $\text{sim}_h(j)$  as the mean cosine similarity of stride- $2^{h-1}$  pairs in  $K_j$ 
        if  $\text{sim}_h(j) > \tau_s^{(h)}$  then
             $L_j \leftarrow \max(L_j, h)$ 
        end if
    end for
end for
return  $L$ 

```

Table 6. Effect of the cosine similarity constraint on Video-MME. All sparse variants are matched at ~ 0.65 FLOP-equivalent sparsity.

Method	Short	Medium	Long	Overall
FA2 (dense)	0.752	0.663	0.537	0.651
PSA w/o sim	0.747	0.667	0.529	0.647
PSA w/ sim	0.748	0.673	0.542	0.654

Unless otherwise noted, all video generation experiments in this appendix are performed on Wan2.1-1.3B at 480p under the same training-free setup as in the main paper, with sparse attention enabled after the first 25% of sampling steps and sparsity reported as FLOP-equivalent sparsity relative to full attention.

C. Additional Ablations

C.1. Cosine Similarity (Video Understanding)

We first evaluate the cosine similarity-based pooling constraint in the video understanding setting. Table 6 compares a dense FlashAttention2 baseline with two PSA variants on Video-MME, where all sparse variants are matched at approximately 0.65 FLOP-equivalent sparsity.

Here, *PSA w/o sim* uses only the importance-based multi-level mask M , while *PSA w/ sim* additionally applies the similarity-based cap L from Alg. 3. Under the same sparsity budget, enabling the constraint improves the overall Video-MME score from 0.647 to 0.654.

C.2. Cosine Similarity (Video Generation)

We next investigate the cosine similarity constraint in the video generation setting. Table 7 compares three presets that share the same PSA framework but differ in how multi-level pooling is assigned and how the similarity thresholds are used.

PSA (sim). This preset adopts a more aggressive importance-based allocation in order to maintain a FLOP-equivalent sparsity close to 0.8 once the similarity cap is applied. Specifically, the thresholds T in mask generation are set to $T = \{0.5, 0.7, 0.8, 0.9\}$, together with similarity thresholds $T_s = \{0.7, 0.65, 0.6\}$. Since the cosine constraint may locally force heterogeneous KV blocks to revert to finer levels, this more progressive allocation ensures that the global sparsity remains in the target range while still avoiding the assignment of coarse pooling to blocks whose neighboring tokens are not sufficiently similar.

PSA (no-sim). To isolate the effect of the similarity constraint, we use a more conservative importance-based assignment with thresholds $T = \{0.7, 0.8, 0.9, 0.9\}$, but disable the cosine cap by setting $T_s = \{-1, -1, -1\}$. This keeps a larger fraction of blocks at the finest level $h = 1$ and slightly reduces sparsity (0.79 vs. 0.80), yet all blocks are assigned purely based on importance scores without checking whether a given KV block is internally homogeneous enough to tolerate

Table 7. Ablation on the cosine similarity constraint for video generation (Wan2.1-1.3B, 480p). All presets operate at comparable FLOP-equivalent sparsity.

Preset	Sparsity \uparrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
PSA (sim)	0.80	23.71	0.8615	0.1086
PSA (no-sim)	0.79	23.36	0.8556	0.1186
PSA (1-level)	0.71	24.42	0.8787	0.0960

Table 8. Ablation on block size and importance pooling for video generation (Wan2.1-1.3B, 480p). All presets operate at an effective sparsity of 0.8.

(b_q, b_k)	Pool	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
(128, 128)	Max	21.43	0.8475	0.1131
(128, 128)	Mean	21.08	0.8337	0.1257
(32, 32)	Max	21.64	0.8529	0.1102
(32, 32)	Mean	21.71	0.8515	0.1101
(64, 64)	Max	21.55	0.8526	0.1099
(64, 64)	Mean	21.56	0.8438	0.1179

high-level pooling.

PSA (1-level). Finally, *PSA (1-level)* uses the same importance-based mask ratios as *PSA (no-sim)*, but sets all similarity thresholds to $T_s = \{1, 1, 1\}$ so that the cosine cap always forces $L_j = 1$ for any kept block. As a result, the combined mask \tilde{M} degenerates to a PSA with $H = 1$, i.e., no pooling is applied and all kept blocks remain at the finest resolution. This configuration therefore has the highest compute cost among the three and serves as an upper bound on reconstruction quality for this importance pattern.

Under comparable sparsity budgets, *PSA (sim)* achieves a better quality-sparsity trade-off than the purely importance-driven *PSA (no-sim)*, indicating that the cosine constraint helps allocate coarse pooling more selectively: it avoids assigning high pyramid levels to KV blocks whose tokens are not poolable, thereby reducing approximation error at fixed sparsity. Meanwhile, *PSA (1-level)* confirms that further quality gains are possible if we completely abandon pooling and pay a significantly higher compute cost, providing a useful reference point for the performance/efficiency envelope of PSA.

C.3. Block Size and Importance Pooling

In this part, we ablate the logical block size (b_q, b_k) and the pooling operator used inside the importance estimation step under a fixed sparsity budget. In PSA’s default formulation (Section 3.3.1), the block importance score is computed as

$$S_{ij} = \max \left(\text{Softmax} \left(\frac{\tilde{Q}_i \tilde{K}_j^\top}{\sqrt{d}} \right) \right),$$

i.e., using max-pooling over the sampled token-pair attention weights. In Table 8, we ablate this choice by replacing the outer “max” with a mean operator:

$$S_{ij}^{\text{mean}} = \text{mean} \left(\text{Softmax} \left(\frac{\tilde{Q}_i \tilde{K}_j^\top}{\sqrt{d}} \right) \right),$$

while keeping every other component (sampling strategy, sparsity, mask-generation rule, and multi-level allocation) unchanged.

At a fixed sparsity of 0.8, PSA is relatively robust to the choice between max- and mean-based importance pooling. Max-pooling yields slightly higher PSNR/SSIM at (128, 128) and (64, 64), while mean-pooling performs comparably at (32, 32). Smaller block sizes—such as (32, 32)—consistently yield better generation quality, as the finer partitioning more precisely isolates high-importance regions in the attention map, reducing approximation errors under the same sparsity budget.

C.4. Multi-Level Allocation Strategy

Finally, we ablate the effect of the multi-level allocation strategy used by the quantile-based method. In this experiment, we fix all other hyper-parameters (block size, overall FLOP-equivalent sparsity, similarity constraint, and sampling strategy)

Table 9. PSA presets: percentage of KV blocks per level. L1/L2/L3 correspond to $h = 1, 2, 3$; Drop corresponds to $h = 0$.

Preset	L1	L2	L3	Drop	Coverage
PSA-1	25%	0%	0%	75%	0.25
PSA-2	0%	0%	100%	0%	1.00
PSA-3	15%	10%	20%	55%	0.45
PSA-4	10%	20%	20%	50%	0.50
PSA-5	10%	10%	40%	40%	0.60

 Table 10. Ablation on the multi-level allocation strategy for PSA on Wan2.1–1.3B, 480p. We vary only how blocks are distributed across pyramid levels (PSA-1 to PSA-5), while keeping the FLOP-equivalent compute budget (about $0.25 \times$ full attention), block size, and all other settings fixed. “KV coverage” denotes the fraction of KV blocks that are kept ($h > 0$).

Preset	KV coverage \uparrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
PSA-1	0.25	20.94	0.8365	0.1286
PSA-2	1.00	14.03	0.3447	0.8445
PSA-3	0.45	22.16	0.8752	0.1004
PSA-4	0.50	22.13	0.8683	0.1045
PSA-5	0.60	21.94	0.8632	0.1070

and vary only how KV blocks are distributed across pyramid levels. All runs are conducted on Wan2.1-1.3B at 480p in the same training-free setting as above, with sparse attention enabled after the first 25% of sampling steps and the compute budget kept at approximately $0.25 \times$ full-attention FLOPs. We report the *KV coverage ratio*, defined as the fraction of KV blocks that are retained (i.e., assigned to levels $h > 0$);

For clarity, we define five *PSA presets*, denoted **PSA-1** to **PSA-5**, which all share the same compute budget but differ in how they allocate KV blocks across the pyramid levels. In this ablation we use three non-dropped levels $h = 1, 2, 3$ (with $h = 1$ the finest and $h = 3$ the coarsest), and a dropped level $h = 0$. The percentage of blocks assigned to each level, together with the resulting KV coverage, is summarized in Table 9.

The quantitative results of this ablation are reported in Table 10. All presets share the same logical block size $(b_q, b_k) = (128, 64)$ and *do not* enable the cosine similarity constraint, so any performance differences can be attributed solely to the multi-level allocation strategy.

Among these variants, PSA-2—which keeps all blocks but collapses them to a single coarse level to stay within the same FLOP budget—leads to severe degradation in all metrics. This shows that, under a fixed compute budget, concentrating almost all computation on one very coarse level yields poor reconstructions—even with maximal KV coverage—because the overly coarse KV representations dominate the performance degradation.

PSA-1, which preserves only a small portion of blocks at the finest representation and drops all others, performs worse than the multi-level strategies. This shows that allocating all compute to a few fine blocks, without using intermediate levels, severely restricts the effective receptive field for each query block and thus leads to degraded performance.

In contrast, the more balanced multi-level allocations in PSA-3/4/5 achieve consistently higher PSNR/SSIM and lower LPIPS at the same FLOP-equivalent cost. These strategies jointly trade off where to keep fine-grained representations and how much area to cover with coarser representations, instead of over-committing to a single level. PSA-3 offers the best overall trade-off and is therefore adopted as our default multi-level strategy in the main video generation experiments. This ablation supports our design choice that, under a fixed compute budget, extremely skewed mask distributions (either too many dropped blocks or too many heavily pooled blocks) hurt quality, whereas a balanced allocation across multiple pyramid levels yields superior performance at similar sparsity. Determining the *optimal* allocation strategy, however, remains an open question and is a promising direction for further exploration.

D. Additional Visual Comparisons

This section provides additional qualitative comparisons for Section 4.2. Each figure follows the same layout as the examples in the main paper.



Figure 5. A plane takes off over a city skyline; cut to a graffiti-covered steam train pulling into a station with billowing smoke. Cinematic colors and motion.

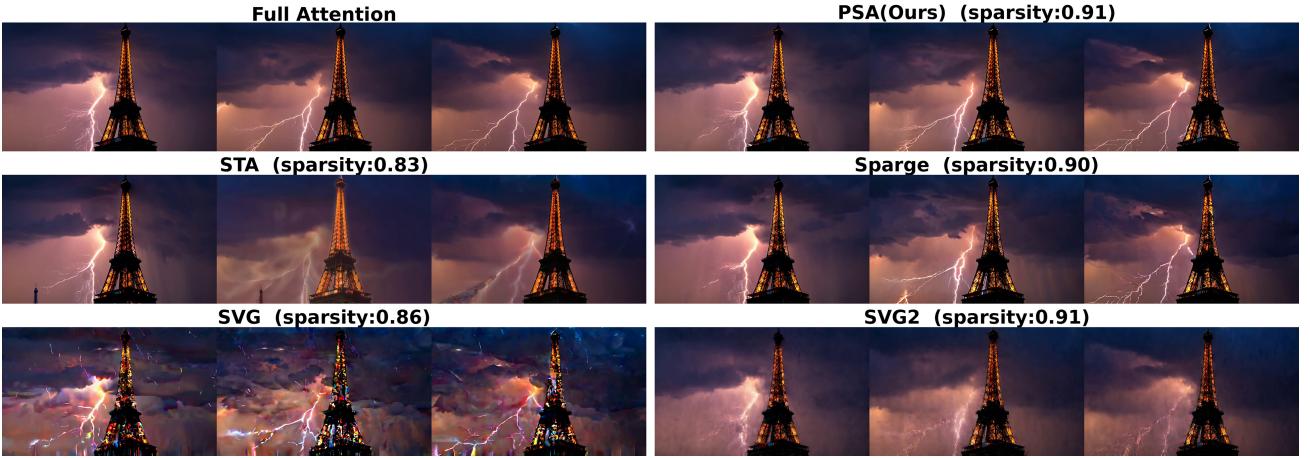


Figure 6. A lightning bolt strikes the Eiffel Tower, illuminating its metal frame against swirling dark storm clouds. A low-angle view highlights the tower's grandeur as the electric flash casts dramatic shadows.



Figure 7. A vintage school bus with retro decals turns a dusty rural corner at sunset. Warm golden light fills the scene as children inside read and play, and the focused driver steers through the tight bend. Low-angle, nostalgic cinematography with rolling hills in the background.



Figure 8. A rider on a sleek black motorcycle weaves through neon-lit city streets at night, wearing a leather jacket and helmet. Dynamic shots follow their smooth maneuvers through traffic, with vibrant signs and skyscrapers creating an intense urban atmosphere.



Figure 9. A cheerful hot dog vendor pushes a colorful cart down a sunny pastel street as a smiling woman in a floral dress chats with passersby. Balloons, flowers, and lively vendors create a bright, carefree vibe.

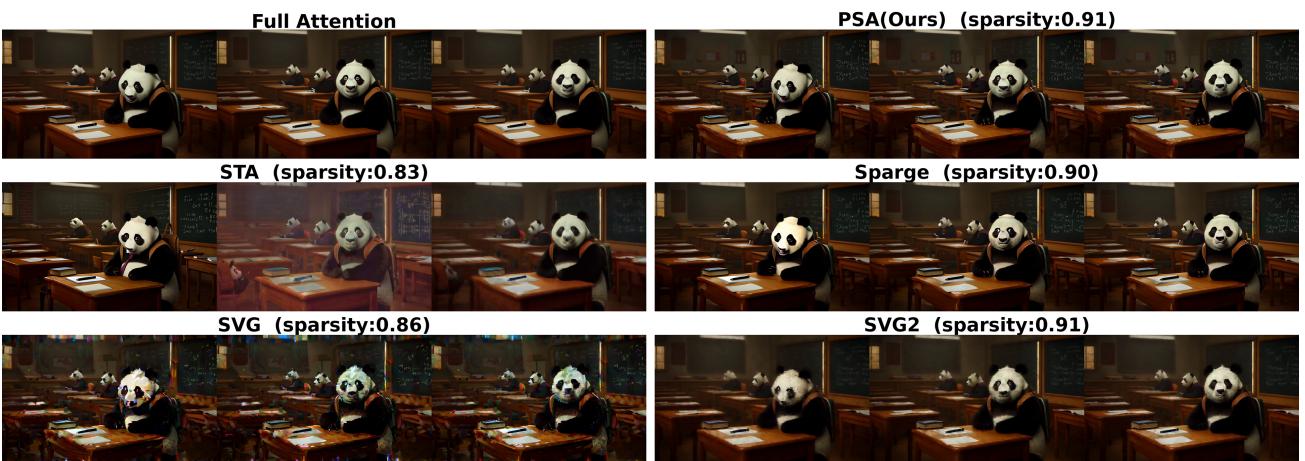


Figure 10. A puzzled panda student with a calculus book in a warm, dimly lit classroom, surrounded by desks, books, and attentive classmates.