MC-SPARSE: DECONSTRUCTING AND CLOSING THE DENSE–SPARSE ATTENTION GAP IN DIFFUSION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse attention is a primary approach to reducing the latency of diffusion transformers in long-sequence generation tasks, such as video and high-resolution 3D asset generation. However, existing methods can degrade generation quality and fidelity at high sparsity levels. Through controlled oracle comparisons, we trace this degradation to three sources: constraints imposed by token grouping, inaccurate interaction selection, and the attention contributions lost when tokens are discarded. Guided by this analysis, we propose Meta-Cached Sparse Attention (MC-Sparse), a training-free framework that selects individual key-value (KV) tokens while organizing similar queries into tile-aligned groups for efficient GPU execution. MC-Sparse caches metadata comprising query groups, KV indices selected using exact attention probabilities, and residuals between dense and sparse attention outputs, and reuses them across subsequent denoising steps. Across video and 3D generation models, MC-Sparse achieves higher fidelity to dense-attention outputs and larger denoising speedups than existing sparse-attention baselines, without visible quality degradation. Relative to dense attention, it delivers a $1.80\times$ denoising speedup on Minimax-H3-Base and a $2.32\times$ speedup on 3D asset generation, both with negligible quality loss.

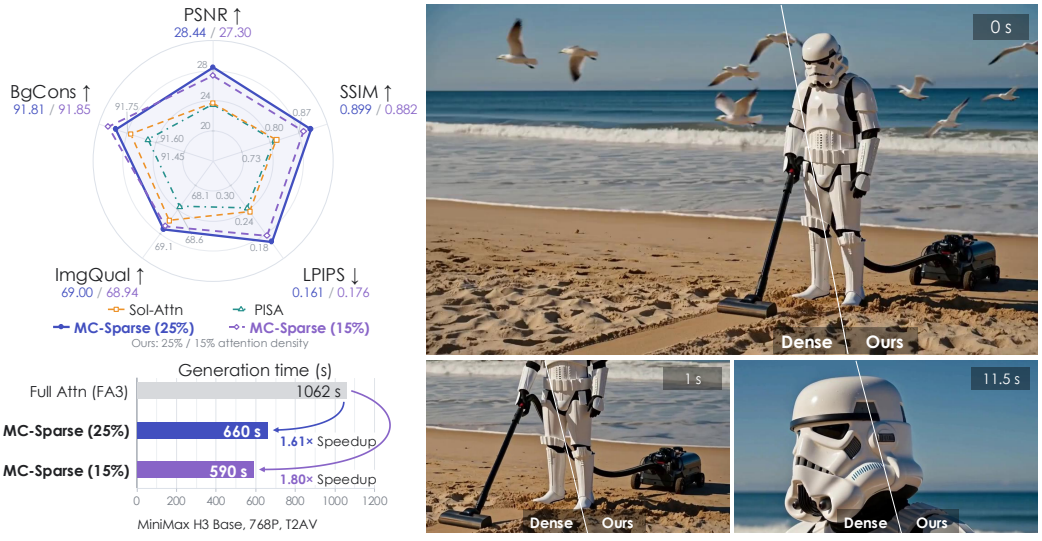

Figure 1: Quality and efficiency of MC-Sparse on Minimax-H3-Base at 768p. Left: quality metrics and DiT denoising latency. At 15% and 25% attention density, MC-Sparse achieves $1.80\times$ and $1.61\times$ speedup over dense attention (FA3), respectively. Right: frames generated at 15% density closely match the dense-attention reference across the video.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Diffusion transformers (Peebles & Xie, 2023) have become a dominant backbone for high-fidelity video and 3D asset generation(Wan et al., 2025; MiniMax, 2026; Lai et al., 2025b;a). However, their growing sequence lengths make attention a major inference bottleneck: its quadratic cost scales rapidly with video duration and output resolution, since it must process spatiotemporal video tokens and geometric 3D tokens. Sparse attention offers a natural remedy, as attention often concentrates on a small subset of token interactions. Yet aggressive sparsity frequently degrades generation quality, leaving a persistent gap between the efficiency of sparse attention and the fidelity of dense attention.

Most existing methods impose sparsity at block granularity (Xi et al., 2025; Zhang et al., 2025c). Block-sparse attention (BSA) selects entire KV blocks for each group of queries, which maps well onto efficient GPU tiles. In practice, block importance is typically estimated from pooled query and key representations(Li et al., 2026b;a). This design entangles two sources of error: the approximate scores used to pick blocks, and the coarse block layout that restricts which interactions can be picked at all. As a result, it is hard to tell how much of the quality gap stems from the selector, how much from the block structure, and how much persists even when both are improved.

We examine these questions through controlled oracle comparisons at matched attention densities (i.e., the same sparsity level). Each oracle maximizes the retained attention mass using exact dense-attention probabilities, but differs in selection granularity: the *block oracle* selects oracle KV blocks for each query group; the *token oracle* selects or oracle individual KV tokens for each query group; and the *per-query oracle* further lets every query choose its own KV tokens. Comparing mean-pooled BSA with these oracles, and the oracles with one another, isolates three sources of the quality gap (Fig. 2):*structural binding error*, which arises because KV tokens are bound together in blocks and different queries in the same group share a single selection; *selection error*, which arises from approximate scoring (e.g. mean pooling); and *discarded tail error*, which arises because omitted tokens still carry nonzero attention mass. These findings motivate three design principles: finer-grained selection, accurate attention scoring, and compensation for discarded contributions.

Realizing these principles efficiently is challenging. First, exact token-level selection requires computing attention scores, which is expensive, and gathering individual KV tokens causes irregular memory access. Second, queries must still be grouped into sufficiently large tiles for efficient GPU execution. Unconstrained clustering can group similar queries to reduce structural binding, but it yields variable-size groups that must be padded to fill the kernel's computation tiles, wasting computation (Zhou et al., 2026). Third, token-level selection removes the block structure that would otherwise provide a natural way to summarize discarded regions for compensation.

Temporal redundancy across denoising steps offers a way to amortize exact selection and to support compensation. We find that reusing exact selections from earlier steps can outperform recomputing mean-pooled selections at every step, and that the dense–sparse output residual changes little between nearby steps. Building on these observations, we propose **Meta-Cached Sparse Attention (MC-Sparse)**, a training-free framework that combines token-level KV selection, tile-aligned query grouping, and cross-step reuse. At anchor steps, MC-Sparse groups similar queries into equal-size tiles, selects KV tokens according to their exact attention mass aggregated over each query group, and records the dense–sparse output residual. Subsequent steps reuse the query groups, selected indices, and cached residuals, while computing attention with the current queries, keys, and values. This amortizes the cost of grouping and exact selection, and the cached residuals compensate for the discarded tail across steps. To make this efficient in practice, we develop fast Principal Direction Divisive Partitioning(PDDP)(Boley, 1998) for grouping, a two-pass exact selector for selection, and a token-sparse attention kernel for execution.

Across the evaluated video models, MC-Sparse achieves higher fidelity to dense-attention outputs and larger denoising speedups than the compared sparse-attention baselines, while preserving visual quality. On Minimax-H3-Base, at 15% attention density, it reaches a $1.80\times$ DiT denoising speedup over dense attention with higher fidelity than competing sparse methods. On HY3D-Internal, at the same density, it achieves a $2.32\times$ denoising speedup with negligible quality degradation.

Our contributions are summarized as follows:

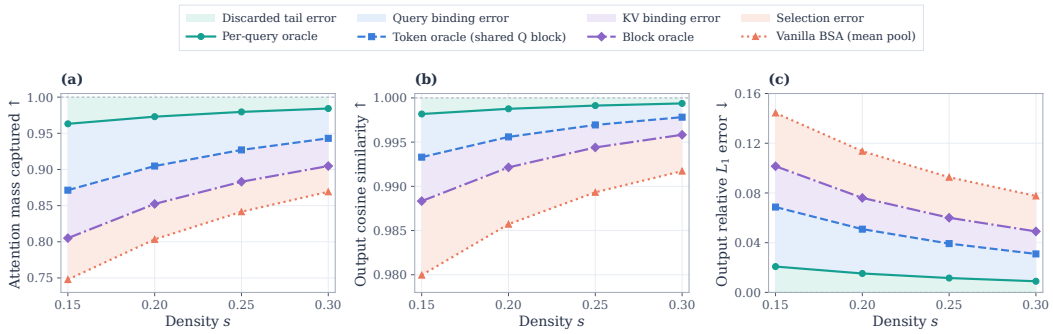

Figure 2: Oracle comparisons of the dense-to-sparse gap on Wan2.1-1.3B-T2V at 480p. Experimental settings are provided in the appendix C

- We use controlled oracle comparisons to disentangle three sources of quality degradation in sparse attention: structural binding error, selection error, and discarded tail error, and quantify how each affects attention fidelity at matched attention densities.
- We propose MC-Sparse, a training-free framework that combines token-level KV selection and tile-aligned query grouping with cross-step reuse of exact selections and output residuals, together with efficient GPU implementations.
- We demonstrate the effectiveness and generality of MC-Sparse on multiple diffusion transformer models, spanning both video and 3D asset generation.

2 Related Work

2.1 Long-Sequence Diffusion Transformers

Diffusion transformers (DiTs) (Peebles & Xie, 2023) are widely used in content generation, with video and 3D asset generation representing two prominent long-sequence tasks. Representative models include HunyuanVideo (Kong et al., 2025), Wan2.1 (Wan et al., 2025), and MiniMax-H3 (MiniMax, 2026) for video generation, and Hunyuan3D 2.5 (Lai et al., 2025a), LATTICE (Lai et al., 2025b), and TRELLIS.2 (Xiang et al., 2025) for 3D asset generation. As generation resolution increases, these tasks require longer token sequences, which can reach tens to hundreds of thousands of tokens, making the quadratic cost of full attention a dominant contributor to inference latency.

2.2 Efficient Attention

Trainable efficient attention. Trainable approaches include linear attention methods such as SANA (Xie et al., 2024), SANA-Video (Chen et al., 2025), and MHLA (Zhang et al., 2026c), and sparse attention methods such as VMoBA (Wu et al., 2025), VSA (Zhang et al., 2025c), and SpargeAttention2 (Zhang et al., 2026a), as well as hybrid attention methods such as SLA (Zhang et al., 2025a) and SLA2 (Zhang et al., 2026b).

Training-free sparse attention. Training-free sparse attention instead aims to approximate full attention in pretrained models while evaluating fewer token interactions. Many approaches organize computation into blocks to exploit FlashAttention-style tiling (Dao et al., 2022). Block selection uses structured patterns or content-dependent importance estimates, as explored by SVG (Xi et al., 2025), XAttention (Xu et al., 2025), and SpargeAttention (Zhang et al., 2025b). SVG-EAR (Zhou et al., 2026) and PISA (Li et al., 2026b) further approximate contributions from unselected blocks. Block construction commonly forms tile-aligned groups based on spatial or spatiotemporal locality, or after serialization along Hilbert or Z-order curves, with SpargeAttention using Hilbert ordering; semantic clustering in SVG2 (Yang et al., 2026), SVOO (Luo et al., 2026), and AdaCluster (Tan et al., 2026) instead produces variable-sized groups whose mismatch with kernel tile boundaries introduces wasted computation within partially filled tiles, reducing actual execution efficiency even without explicit padding. A complementary line of work exploits cross-step redundancy in attention, a strategy also used in diffusion caching (Wang et al., 2026a;b). AdaSpa (Xia et al., 2025), DiTFastAttn (Yuan et al., 2024), and QuantSparse (Feng et al., 2026) reuse attention-related information along the denoising trajectory.

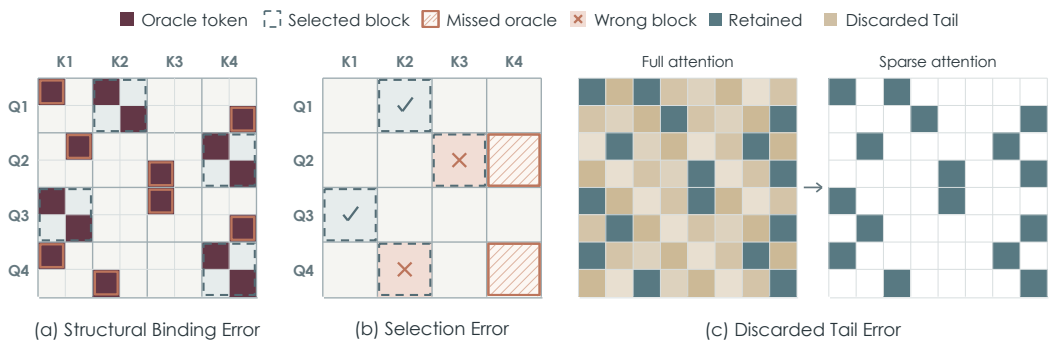

Figure 3: Three sources of the gap between dense and sparse attention.

3 PRELIMINARIES

Dense attention. Diffusion transformers typically use multi-head attention. For a single head with query, key, and value matrices $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, and query/key dimension d, dense attention computes $\mathbf{O}^{\text{full}} = \text{Softmax}(\mathbf{QK}^\top/\sqrt{d})\mathbf{V}$, where the softmax is taken over keys. We write A_{ij} for the resulting attention weight from query i to key j.

Block-sparse attention. Sparse attention restricts each query to a subset of keys: $\mathbf{O}^{\text{sparse}} = \text{Softmax}(\mathbf{QK}^\top/\sqrt{d} + \mathbf{M})\mathbf{V}$, where $M_{ij} = 0$ retains the query–key interaction and $M_{ij} = -\infty$ masks it out. For efficient GPU execution, block-sparse attention (BSA) partitions queries and keys into blocks, retains entire KV blocks, and shares one selection among all queries in a block. The block size is typically aligned with the GPU computation tile.

Vanilla BSA. We refer to BSA with a mean-pooled selector as *vanilla BSA*. At each diffusion step, this selector scores each query-block/KV-block pair (g, r) by $\hat{z}_{gr} = \bar{\mathbf{q}}_g^\top \bar{\mathbf{k}}_r/\sqrt{d}$, where $\bar{\mathbf{q}}_g$ and $\bar{\mathbf{k}}_r$ are the mean-pooled query and key of blocks g and r, respectively. For each query block, it retains the highest-scoring KV blocks under a given attention density.

4 METHOD

We first examine where vanilla BSA loses fidelity relative to dense attention. An *oracle selector* retains the entries with the largest dense-attention probability mass under a fixed attention density. We consider three oracles (Fig. 2): the *block oracle* selects KV blocks for each query block; the *token oracle (shared Q block)* selects individual KV tokens but shares one selection within each query block; and the *per-query oracle* selects KV tokens independently for each query. At matched densities, comparing vanilla BSA with these oracles and with dense attention isolates three sources of the gap: structural binding error, selection error, and discarded tail error.

Because exact selection and query grouping are costly, MC-Sparse exploits the stability of attention across denoising steps: it performs them only at *anchor steps* and reuses the resulting query groups, KV indices, and dense–sparse output residual at subsequent *reuse steps*. In the following subsections, we analyze each error in turn and introduce a corresponding design.

4.1 MITIGATING STRUCTURAL BINDING ERROR

Structural binding error arises from two constraints of block-sparse attention. *KV binding* forces a block-sparse selector to retain irrelevant neighbors of important tokens, while *query binding* forces queries with different attention patterns to share one selection (Fig. 3(a)). Moving from the block oracle to the token oracle with shared query blocks removes KV binding, and moving further to the per-query oracle removes query binding. The accuracy gains at each step (Fig. 2) show that both constraints cause losses. Table 4 further confirms this at matched realized densities: finer KV granularity and grouping more similar queries both improve fidelity.

Token-level KV selection. To remove KV binding, MC-Sparse selects individual KV tokens for each query group across block boundaries. The token level sparse attention kernel gathers the selected keys and values online from their indices without repacking them into blocks.

Tile-aligned query grouping. With head-specific keys and values in MHA, efficient attention tiles must batch queries within each head. We therefore form groups $\mathcal{G}_g$ of C similar queries that share one KV selection, with C matched to the kernel's query tile size. Minimizing within-group variance of $\mathbf{Q}$ is equivalent to maximizing between-group variance. Standard clustering methods such as k-means do not guarantee equal-size groups, while enforcing hard balance constraints adds assignment overhead and complicates GPU parallelization. We adopt a median-split version of principal direction divisive partitioning (PDDP) (Boley, 1998): it recursively projects each group onto its principal direction and splits the ordered queries at the median until every group contains exactly C queries, promoting within-group similarity while maintaining tile alignment. Because a naive PDDP implementation is expensive on GPUs, we develop Fast PDDP as described in Section 4.4.

4.2 Mitigating Selection Error

For a query group $\mathcal{G}_g$ and a budget of K KV tokens, we define *oracle selection* as the set that maximizes the group's cumulative dense-attention mass:

$$\mathcal{S}_g^* = \arg\max_{|\mathcal{S}|=K} \sum_{i\in\mathcal{G}_g} \sum_{j\in\mathcal{S}} A_{ij}. \tag{1}$$

Here $\mathcal{S}$ is a set of KV indices. This objective is motivated by an output-error bound proportional to discarded attention mass for bounded value norms (Shen et al., 2026). Vanilla BSA instead scores pooled representations; even pooling queries alone does not preserve average attention probabilities:

$$\mathrm{Softmax}(\bar{\mathbf{q}}_g \mathbf{K}^\top/\sqrt{d})_j \neq \frac{1}{|\mathcal{G}_g|} \sum_{i\in\mathcal{G}_g} A_{ij}, \tag{2}$$

where $\bar{\mathbf{q}}_g$ is the group mean. Pooled scoring can miss KV tokens with high cumulative mass.

Reusable exact KV Selection. Without a trained indexer, recomputing exact attention scores at every step is costly. Attention patterns remain stable across nearby denoising steps (Xia et al., 2025); exact selections from earlier steps outperform mean-pooled selection recomputed at every step (Fig. 4). This supports computing Eq. 1 only at anchor steps and reusing the result. At step τ, we obtain $\mathcal{S}_{\tau,g}$ by selecting the top-K tokens ranked by $\sum_{i\in\mathcal{G}_{\tau,g}} A_{\tau,ij}$. We cache the query groups $\mathcal{G}_\tau$ and selected indices $\mathcal{S}_\tau$ for subsequent steps, amortizing both grouping and selection. The current $\mathbf{Q}_t$, $\mathbf{K}_t$, and $\mathbf{V}_t$ are recomputed at each step.

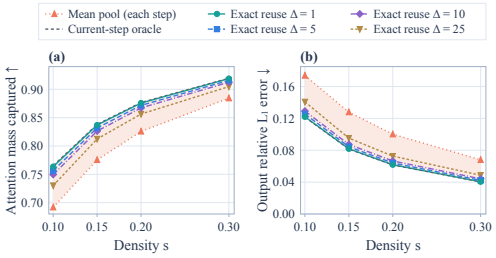

Figure 4: Temporal stability and accuracy of reused exact selection.

4.3 Mitigating Discarded Tail Error

Discarded tokens typically have small attention weights but still make nonzero contributions (Fig. 3(c)), leaving an output gap to dense attention even with per-query oracle selection.

Temporal residual compensation. Block-sparse methods approximate discarded contributions by incorporating block-mean representations into attention or correcting outputs with block-level statistics (Li et al., 2026b; Zhou et al., 2026). Token-level selection cuts irregularly across these blocks, preventing direct reuse of their summaries. We instead exploit the temporal dimension: the dense–sparse residual varies less across denoising steps than the attention output, with exact selection further reducing this variation compared with mean-pooled selection (Fig. 8).

We use a simple additive residual correction (Yuan et al., 2024). At anchor step τ, we record

$$\mathbf{R}_\tau = \mathbf{O}_\tau^{\text{full}} - \mathbf{O}_\tau^{\text{sparse}}, \tag{3}$$

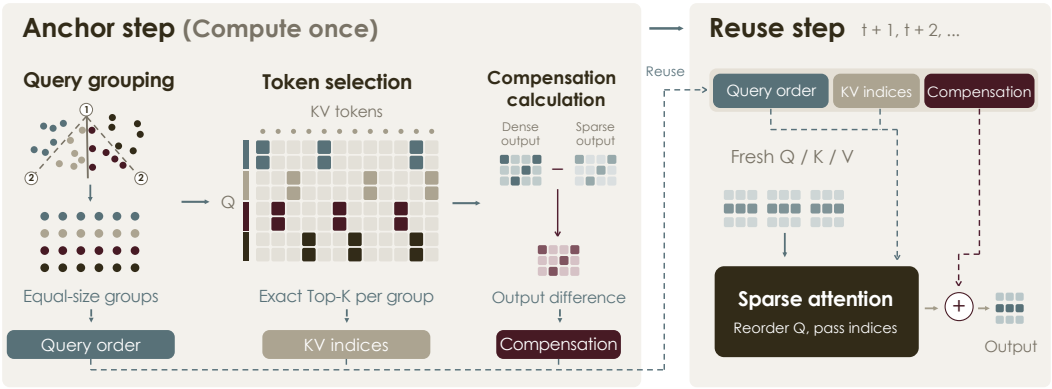

Figure 5: Pipeline overview of MC-Sparse.

where $\mathbf{O}_\tau^{\text{sparse}}$ uses the exact KV selection. At a subsequent reuse step t, MC-Sparse computes

$$\widehat{\mathbf{O}}_t = \text{SparseAttn}\left(\mathbf{Q}_t, \mathbf{K}_t, \mathbf{V}_t; \mathcal{G}_\tau, \mathcal{S}_\tau\right) + \mathbf{R}_\tau. \tag{4}$$

The residual follows the same refresh schedule as the cached groups and KV indices. Because the anchor step already computes dense attention for exact selection, forming the residual adds one sparse-attention evaluation and an elementwise subtraction at each anchor.

4.4 Pipeline and Efficient Implementation

Overall pipeline. MC-Sparse alternates between anchor and reuse steps. At an anchor step, which is the last step of a dense warm-up or refresh segment, we run dense attention, construct query groups, perform exact KV selection, and cache the query groups, selected indices, and output residual. At each subsequent reuse step, the current $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are recomputed, while attention uses the cached metadata and residual. Figure 5 illustrates the pipeline; the pseudocode is provided in Appendix B.

Fast PDDP. We obtain principal directions using batched power iteration, processing all nodes at the same tree depth across attention heads in parallel. The implementation avoids materializing covariance matrices and only permutes query-index arrays between tree levels. Table 5 reports its single-call and amortized grouping costs.

Two-pass exact selection. FlashAttention does not expose attention logits, so we implement a two-pass kernel for exact KV selection. Details are provided in Appendix C.2.

Token-sparse attention kernel. We implement the token-sparse attention kernel in CuTeDSL, with performance comparisons in Table 6 and design details in Appendix C.2.

Cache offloading and prefetching. When GPU memory is insufficient, we offload inactive cache entries to CPU memory and prefetch them ahead of use, overlapping transfers with computation to hide latency. Table 11 demonstrates the effectiveness of prefetching.

5 Experiments

5.1 Experimental Setup

Tasks and baselines. We evaluate video generation with Minimax-H3-Base (MiniMax, 2026), HunyuanVideo-13B (Kong et al., 2025), and Wan2.1-14B-T2V/I2V (Wan et al., 2025), and image-to-geometry generation with HY3D-Internal. Following SVG-EAR (Zhou et al., 2026), we use Penguin Benchmark prompts for text-to-video and VBench image–prompt pairs for image-to-video, and a detail-oriented image test set for 3D generation. We compare with FA3-based dense attention (Shah et al., 2024), SVG2, SVG-EAR, Sol-Attn (Li et al., 2026a), PISA (Li et al., 2026b), and SpargeAttn (Zhang et al., 2025b), as applicable to each model.

Metrics. PSNR, SSIM, and LPIPS (Zhang et al., 2018) measure video fidelity to dense-attention outputs, while VBench (Huang et al., 2023) measures image quality (ImgQual) and background consistency (BgCons). For 3D assets, we report geometric fidelity using Chamfer distance (CD), vol-

Table 1: Video generation results. PSNR, SSIM, and LPIPS compare with dense-attention outputs; ImgQual and BgCons are VBench scores. Speedup measures DiT denoising only. Bold and underlined values indicate the best and second-best results, respectively, excluding Full Attn.

Method	PSNR↑	SSIM↑	LPIPS↓	ImgQual↑	BgCons↑	Density↓	Speedup↑
Minimax-H3-Base, 768p, T2AV (video branch)							
Full Attn (FA3)	-	-	-	69.20	91.72	100%	1.00×
Sol-Attn	23.66	0.816	0.234	68.83	91.73	32.4%	1.59×
PISA	23.45	0.811	0.243	68.54	91.64	30.0%	1.52×
MC-Sparse (density=25%)	**28.44**	**0.899**	**0.161**	**69.00**	91.81	25.0%	1.61×
MC-Sparse (density=15%)	27.30	0.882	0.176	68.94	**91.85**	15.0%	**1.80×**
HunyuanVideo-13B, 720p, T2V							
Full Attn (FA3)	-	-	-	68.52	96.91	100%	1.00×
Sol-Attn	28.06	0.898	0.159	68.27	**97.03**	32.3%	1.79×
PISA	27.65	0.890	0.174	**68.74**	96.96	30.0%	1.70×
MC-Sparse (density=25%)	**32.89**	**0.940**	**0.114**	68.66	96.89	25.0%	1.82×
MC-Sparse (density=15%)	30.78	0.919	0.135	68.67	96.96	15.0%	**2.00×**
Wan2.1-14B-T2V, 720p							
Full Attn (FA3)	-	-	-	67.11	96.60	100%	1.00×
SpargeAttn	23.56	0.828	0.212	66.84	**96.81**	30.0%	-
Sol-Attn	24.62	0.852	0.157	67.01	96.72	31.6%	1.52×
PISA	24.03	0.838	0.202	67.09	96.76	30.0%	1.52×
SVG2	26.02	0.875	0.166	67.11	96.52	31.5%	1.34×
SVG-EAR	27.61	0.897	0.142	66.97	96.41	25.3%	1.38×
MC-Sparse (Ours)	**28.81**	**0.912**	**0.128**	**67.12**	96.70	25.0%	**1.53×**
Wan2.1-14B-I2V, 720p							
Full Attn (FA3)	-	-	-	70.37	96.51	100%	1.00×
SpargeAttn	26.66	0.859	0.172	70.40	96.30	30.0%	-
Sol-Attn	27.73	0.876	0.157	70.34	**96.46**	31.9%	1.51×
PISA	27.10	0.865	0.163	70.38	96.40	30.0%	1.51×
SVG2	27.15	0.861	0.168	70.32	96.42	30.4%	1.35×
SVG-EAR	30.71	0.916	0.127	70.34	96.38	24.6%	1.39×
MC-Sparse (Ours)	**32.11**	**0.929**	**0.117**	**70.41**	**96.46**	25.0%	**1.52×**

umetric IoU, and F1, and input-image consistency using Uni3D-I (Zhou et al., 2023) and ULIP3D-I (Xue et al., 2023). Efficiency is measured by attention density, the fraction of retained query–key interactions, and DiT denoising speedup over dense attention, including warm-up but excluding condition encoding and VAE decoding. Detailed settings are provided in Appendix C.

5.2 Main Results

Video generation. Table 1 shows improved fidelity and denoising speedup across the evaluated video models. On Minimax-H3-Base, MC-Sparse achieves 28.44 dB PSNR at 25% density and 1.61× speedup, compared with 23.66 dB and 1.59× for Sol-Attn at 32.4% density. With the same anchor schedule, MC-Sparse-Flash reaches 1.80× speedup at 15% density with 27.30 dB PSNR, retaining higher fidelity than both Sol-Attn and PISA. On Hunyuan and Wan, MC-Sparse achieves the highest PSNR and SSIM and the lowest LPIPS among the evaluated sparse methods, with 1.52×–1.82× speedup. The qualitative comparisons in Fig. 6 show that MC-Sparse produces videos visually closer to the dense-attention outputs than the compared sparse methods, consistent with the fidelity metrics. Qualitative results on other video generation models are provided in Fig. 9.

3D asset generation. At 15% attention density, MC-Sparse achieves 2.32× denoising speedup and the best geometric fidelity among the evaluated sparse methods (Table 2), reducing CD from PISA's 0.976 to 0.177. Its input-image consistency remains comparable to dense attention. In Fig. 7, MC-Sparse better preserves fine details and surface continuity, while the compared methods exhibit broken surfaces and missing details despite using higher attention densities.

5.3 Ablation Study

Component contributions. Starting from vanilla BSA, we successively add exact KV selection with cross-step reuse, token-level KV granularity, query grouping, and residual compensation (Ta-



Figure 6: Qualitative comparison of video generation using Minimax-H3-Base.

Table 2: Image-to-geometry results on HY3D-Internal. Geometric fidelity is measured against dense-attention outputs; Uni3D-I and ULIP3D-I measure input-image consistency. Speedup measures DiT denoising only.

Config	CD↓	Vol-IoU-1536↑	F1@0.001↑	Uni3D-I↑	ULIP3D-I↑	Density↓	Speedup↑
HY3D-Internal							
Full Attn (FA3)	-	-	-	0.3309	0.1204	100%	1.00×
Sol-Attn	1.901	50.55	70.36	0.3301	0.1203	36.1%	1.57×
PISA	0.976	63.22	83.29	0.3304	0.1204	25.0%	$<$ 1.87×
MC-Sparse (Ours)	**0.177**	**82.91**	**96.33**	**0.3311**	**0.1206**	15.0%	**2.32**×

ble 3). Each addition improves PSNR, SSIM, and LPIPS across all evaluated densities, supporting the contribution of each component to the full framework.

KV granularity and query grouping. Table 4 shows that token-level KV selection improves attention recall and output accuracy under every grouping strategy. We match realized density after tile alignment, since padding variable-size k-means groups increases executed interactions beyond the nominal budget. Under this comparison, Fast PDDP avoids alignment inflation and consistently outperforms k-means, supporting both token-level selection and tile-aligned query grouping.

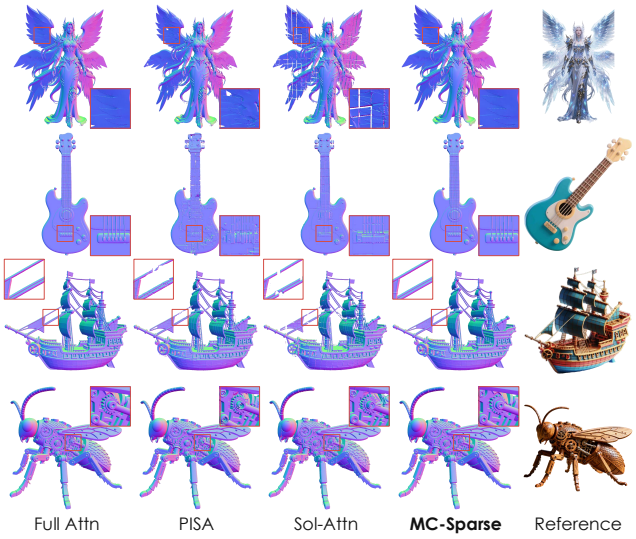

Figure 7: Qualitative comparison on 3D asset generation.

Role of residual compensation. Residual compensation yields the largest gain in the cumulative ablation, but its benefit is substantially greater with MC-Sparse's selection and grouping designs than with vanilla BSA (Table 10), indicating that its effectiveness depends on their combination. PISA receives smaller additional gains than vanilla BSA with its native block-statistics compensation retained, consistent with both mechanisms compensating for discarded tail contributions.

5.4 System Efficiency

Token-sparse attention kernel. Table 6 compares attention kernels at matched KV-token densities on Wan2.1-14B-T2V at 720p, excluding selection and grouping. SVG2 uses $Q_c = 300$ and $K_c =$

Table 3: Cumulative ablation on Wan2.1-1.3B-T2V at 480p. Each row adds one component to the preceding configuration. Settings are provided in Appendix C.

Configuration	Density $s = 0.2$			Density $s = 0.25$			Density $s = 0.3$		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Vanilla BSA	20.96	0.7535	0.2714	21.67	0.7750	0.2469	22.26	0.7923	0.2278
+ exact KV selection (w. reuse)	21.60	0.7731	0.2482	22.63	0.8004	0.2176	23.47	0.8205	0.1956
+ token granularity	22.57	0.7954	0.2254	23.54	0.8188	0.1995	24.38	0.8380	0.1789
+ query grouping	23.51	0.8175	0.2017	24.47	0.8386	0.1784	25.24	0.8539	0.1624
+ compensation (full model)	**27.05**	**0.8809**	**0.1326**	**27.35**	**0.8843**	**0.1294**	**27.54**	**0.8862**	**0.1273**

Table 4: Query grouping and KV granularity at matched realized density $s = 0.20$. Align. is the executed-to-nominal interaction ratio. Full results appear in Table 9 in the appendix.

Grouping	KV	Align.	Recall ↑	Rel. L_1 ↓
No reorder	block	1.00	0.778	0.101
	token	1.00	0.857	0.070
k-means	block	2.08	0.830	0.098
	token	1.45	0.904	0.054
Fast PDDP	block	1.00	0.887	0.062
	token	1.00	**0.923**	**0.042**

Table 5: Query-grouping latency on Wan2.1-T2V (ms per layer). Amortized over 40 steps: 2 Fast PDDP calls or Flash-KMeans with a 50-iteration initialization and 39 two-iteration updates.

	1.3B, 480p		14B, 720p	
Grouping	Single	Amort.	Single	Amort.
Flash-KMeans init.	32.0	2.26	295.1	20.43
+ step update	1.50		13.38	
Fast PDDP	19.5	**0.97**	173.4	**8.67**

Table 6: Attention-kernel speedup on Wan2.1-14B-T2V (720p), using a Hopper GPU in BF16 ($H = 40$, $D = 128$, $S = 75600$). Entries show speedup $\rho = t_{\mathrm{FA3}}/t$ (efficiency $\eta = \rho s$ relative to the ideal $1/s$ speedup). Selection and grouping are excluded.

Kernel	Sparse layout	$s = 0.2$	$s = 0.3$	$s = 0.5$
BSA	regular blocks	4.96× (0.99)	3.30× (0.99)	1.97× (0.99)
PISA kernel	regular blocks	4.11× (0.82)	2.84× (0.85)	1.73× (0.87)
SVG2 kernel	variable-length blocks	4.02× (0.80)	2.70× (0.81)	1.62× (0.81)
MC-Sparse	tile-aligned Q, gathered KV	4.79× (**0.96**)	3.25× (**0.98**)	1.97× (**0.99**)

1000. MC-Sparse achieves 96%–99% of the ideal $1/s$ speedup, versus 80%–81% for SVG2 and 82%–87% for PISA, showing that token-level KV gathering preserves kernel efficiency.

Tile-aligned query grouping. Table 5 compares Fast PDDP with Flash-KMeans. Under the evaluated schedules, Fast PDDP reduces amortized grouping cost by 2.32× at 480p and 2.36× at 720p, keeping tile-aligned query grouping cheep.

Anchor-step overhead. The two-pass selector adds a QK-only pass costing approximately half a dense attention evaluation in GEMM operations. With the first anchor overlapping the final warm-up step (Table 7), the additional dense evaluation and selection passes yield an amortized overhead equivalent to an absolute density increase of $\Delta s \approx 5\%$ over all denoising steps. Including query grouping and other auxiliary operations, the measured overhead is equivalent to $\Delta s \approx 5\%$–6%, below SVG2's combined clustering and variable-length kernel overhead.

6 Conclusion

We presented MC-Sparse, a training-free framework for accelerating diffusion transformers with token-level sparse attention. Our oracle comparisons show that the dense-to-sparse gap extends beyond selection accuracy to include structural binding and discarded tail contributions. Guided by this analysis, MC-Sparse combines token-level KV selection and tile-aligned query grouping with temporal reuse and residual compensation, balancing approximation quality with execution efficiency. Experiments on video and 3D asset generation demonstrate improved fidelity to dense-attention outputs and reduced denoising latency compared with the evaluated sparse-attention baselines.

REFERENCES

Daniel Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2 (4):325–344, 1998. URL citeseer.ist.psu.edu/boley97principal.html.

Junsong Chen, Yuyang Zhao, Jincheng Yu, Ruihang Chu, Junyu Chen, Shuai Yang, Xianbang Wang, Yicheng Pan, Daquan Zhou, Huan Ling, Haozhe Liu, Hongwei Yi, Hao Zhang, Muyang Li, Yukang Chen, Han Cai, Sanja Fidler, Ping Luo, Song Han, and Enze Xie. Sana-video: Efficient video generation with block linear diffusion transformer, 2025. URL https://arxiv.org/abs/2509.24695.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

Weilun Feng, Chuanguang Yang, Haotong Qin, Mingqiang Wu, Yuqi Li, Xiangqi Li, Zhulin An, Libo Huang, Yulun Zhang, Michele Magno, and Yongjun Xu. Quantsparse: Comprehensively compressing video diffusion transformer with model quantization and attention sparsification, 2026. URL https://arxiv.org/abs/2509.23681.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. Vbench: Comprehensive benchmark suite for video generative models, 2023. URL https://arxiv.org/abs/2311.17982.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, Kathrina Wu, Qin Lin, Junkun Yuan, Yanxin Long, Aladdin Wang, Andong Wang, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Hongmei Wang, Jacob Song, Jiawang Bai, Jianbing Wu, Jinbao Xue, Joey Wang, Kai Wang, Mengyang Liu, Pengyu Li, Shuai Li, Weiyan Wang, Wenqing Yu, Xinchi Deng, Yang Li, Yi Chen, Yutao Cui, Yuanbo Peng, Zhentao Yu, Zhiyu He, Zhiyong Xu, Zixiang Zhou, Zunnan Xu, Yangyu Tao, Qinglin Lu, Songtao Liu, Dax Zhou, Hongfa Wang, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, and Caesar Zhong. Hunyuanvideo: A systematic framework for large video generative models, 2025. URL https://arxiv.org/abs/2412.03603.

Zeqiang Lai, Yunfei Zhao, Haolin Liu, Zibo Zhao, Qingxiang Lin, Huiwen Shi, Xianghui Yang, Mingxin Yang, Shuhui Yang, Yifei Feng, Sheng Zhang, Xin Huang, Di Luo, Fan Yang, Fang Yang, Lifu Wang, Sicong Liu, Yixuan Tang, Yulin Cai, Zebin He, Tian Liu, Yuhong Liu, Jie Jiang, Linus, Jingwei Huang, and Chunchao Guo. Hunyuan3d 2.5: Towards high-fidelity 3d assets generation with ultimate details, 2025a. URL https://arxiv.org/abs/2506.16504.

Zeqiang Lai, Yunfei Zhao, Zibo Zhao, Haolin Liu, Qingxiang Lin, Jingwei Huang, Chunchao Guo, and Xiangyu Yue. Lattice: Democratize high-fidelity 3d generation at scale, 2025b. URL https://arxiv.org/abs/2512.03052.

Haopeng Li, Yitong Li, Junsong Chen, Tian Ye, Haozhe Liu, Jincheng Yu, Duomin Wang, Ruihua Zhang, Zeke Xie, Enze Xie, and Song Han. Sol-attn: Accelerating video generation inference via on-the-fly attention sparsification, 2026a. URL https://arxiv.org/abs/2607.24027.

Haopeng Li, Shitong Shao, Wenliang Zhong, Zikai Zhou, Lichen Bai, Hui Xiong, and Zeke Xie. Pisa: Piecewise sparse attention is wiser for efficient diffusion transformers, 2026b. URL https://arxiv.org/abs/2602.01077.

Jiayi Luo, Jiayu Chen, Jiankun Wang, Cong Wang, Hanxin Zhu, Qingyun Sun, Chen Gao, Zhibo Chen, and Jianxin Li. Attention sparsity is input-stable: Training-free sparse attention for video generation via offline sparsity profiling and online qk co-clustering, 2026. URL https://arxiv.org/abs/2603.18636.

MiniMax. MiniMax H3: An Open Model Breaking the Boundaries Between Tasks and Modalities. Official research blog, 2026. URL https://www.minimax.io/blog/minimax-h3. Accessed 2026-09-17.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision, 2024. URL https://arxiv.org/abs/2407.08608.

Zhenyi Shen, Junru Lu, Lin Gui, Jiazheng Li, Yulan He, Di Yin, and Xing Sun. Ssa: Sparse sparse attention by aligning full and sparse attention outputs in feature space, 2026. URL https://arxiv.org/abs/2511.20102.

Haoyue Tan, Shengnan Wang, Yulin Qiao, Juncheng Zhang, Youhui Bai, Ping Gong, Zewen Jin, and Cheng Li. Adacluster: Adaptive query-key clustering for sparse attention in video generation, 2026. URL https://arxiv.org/abs/2604.18348.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025. URL https://arxiv.org/abs/2503.20314.

Jiangshan Wang, Zeqiang Lai, Jiarui Chen, Jiayi Guo, Hang Guo, Xiu Li, Xiangyu Yue, and Chunchao Guo. Elastic diffusion transformer, 2026a. URL https://arxiv.org/abs/2602.13993.

Jiangshan Wang, Kang Zhao, Jiayi Guo, Jiayu Wang, Hang Guo, Chenyang Zhu, Xiu Li, and Xiangyu Yue. Precisecache: Precise feature caching for efficient and high-fidelity video generation, 2026b. URL https://arxiv.org/abs/2603.00976.

Jianzong Wu, Liang Hou, Haotian Yang, Xin Tao, Ye Tian, Pengfei Wan, Di Zhang, and Yunhai Tong. Vmoba: Mixture-of-block attention for video diffusion models, 2025. URL https://arxiv.org/abs/2506.23858.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity, 2025. URL https://arxiv.org/abs/2502.01776.

Yifei Xia, Suhan Ling, Fangcheng Fu, Yujie Wang, Huixia Li, Xuefeng Xiao, and Bin Cui. Training-free and adaptive sparse attention for efficient long video generation, 2025. URL https://arxiv.org/abs/2502.21079.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3d generation, 2025. URL https://arxiv.org/abs/2512.14692.

Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, and Song Han. Sana: Efficient high-resolution image synthesis with linear diffusion transformers, 2024. URL https://arxiv.org/abs/2410.10629.

Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring, 2025. URL https://arxiv.org/abs/2503.16428.

Le Xue, Mingfei Gao, Chen Xing, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding, 2023. URL https://arxiv.org/abs/2212.05171.

Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Jianfei Chen, Song Han, Kurt Keutzer, and Ion Stoica. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation, 2026. URL https://arxiv.org/abs/2505.18875.

Zhihang Yuan, Hanling Zhang, Pu Lu, Xuefei Ning, Linfeng Zhang, Tianchen Zhao, Shengen Yan, Guohao Dai, and Yu Wang. Ditfastattn: Attention compression for diffusion transformer models, 2024. URL https://arxiv.org/abs/2406.08552.

Jintao Zhang, Haoxu Wang, Kai Jiang, Shuo Yang, Kaiwen Zheng, Haocheng Xi, Ziteng Wang, Hongzhou Zhu, Min Zhao, Ion Stoica, Joseph E. Gonzalez, Jun Zhu, and Jianfei Chen. Sla: Beyond sparsity in diffusion transformers via fine-tunable sparse-linear attention, 2025a. URL https://arxiv.org/abs/2509.24006.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattention: Accurate and training-free sparse attention accelerating any model inference, 2025b. URL https://arxiv.org/abs/2502.18137.

Jintao Zhang, Kai Jiang, Chendong Xiang, Weiqi Feng, Yuezhou Hu, Haocheng Xi, Jianfei Chen, and Jun Zhu. Spargeattention2: Trainable sparse attention via hybrid top-k+top-p masking and distillation fine-tuning, 2026a. URL https://arxiv.org/abs/2602.13515.

Jintao Zhang, Haoxu Wang, Kai Jiang, Kaiwen Zheng, Youhe Jiang, Ion Stoica, Jianfei Chen, Jun Zhu, and Joseph E. Gonzalez. Sla2: Sparse-linear attention with learnable routing and qat, 2026b. URL https://arxiv.org/abs/2602.12675.

Kewei Zhang, Ye Huang, Yufan Deng, Jincheng Yu, Junsong Chen, Huan Ling, Enze Xie, and Daquan Zhou. Mhla: Restoring expressivity of linear attention via token-level multi-head, 2026c. URL https://arxiv.org/abs/2601.07832.

Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric Xing, and Hao Zhang. Vsa: Faster video diffusion with trainable sparse attention, 2025c. URL https://arxiv.org/abs/2505.13389.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. URL https://arxiv.org/abs/1801.03924.

Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Uni3d: Exploring unified 3d representation at scale, 2023. URL https://arxiv.org/abs/2310.06773.

Xuanyi Zhou, Qiuyang Mang, Shuo Yang, Haocheng Xi, Jintao Zhang, Huanzhi Mao, Joseph E. Gonzalez, Kurt Keutzer, Ion Stoica, and Alvin Cheung. Svg-ear: Parameter-free linear compensation for sparse video generation via error-aware routing, 2026. URL https://arxiv.org/abs/2603.08982.

A AI USE DISCLOSURE

We used generative AI tools to assist with drafting and editing the manuscript, including feedback on the presentation of the methodology, experiments, and results, and to generate or revise LaTeX, plotting code, figures, and tables. The authors reviewed and verified all AI-assisted content. Generative AI was not used to generate experimental data or model outputs. The authors take full responsibility for the final content of this work.

B MC-SPARSE ALGORITHM

Algorithm 1 summarizes the anchor-and-reuse pipeline described in Section 4.4.

Algorithm 1 MC-Sparse

Require: Diffusion steps T, reuse interval L, KV budget K
Ensure: Attention outputs $\{\widehat{\mathbf{O}}_t\}_{t=1}^{T}$

for $t = 1, \ldots, T$ **do**
  Compute $\mathbf{Q}_t$, $\mathbf{K}_t$, and $\mathbf{V}_t$
  if IsAnchor(t, L) **then**
    $(\mathbf{O}_t^{\text{full}}, \mathbf{l}_t) \leftarrow \text{FlashAttn}(\mathbf{Q}_t, \mathbf{K}_t, \mathbf{V}_t)$
    $\mathcal{G}_t \leftarrow \text{GroupQueries}(\mathbf{Q}_t)$
    $\mathcal{S}_t \leftarrow \text{ExactKVSelection}(\mathbf{Q}_t, \mathbf{K}_t, \mathbf{l}_t, \mathcal{G}_t, K)$
    $\mathbf{O}_t^{\text{sparse}} \leftarrow \text{SparseAttn}(\mathbf{Q}_t, \mathbf{K}_t, \mathbf{V}_t; \mathcal{G}_t, \mathcal{S}_t)$
    $\mathbf{R}_t \leftarrow \mathbf{O}_t^{\text{full}} - \mathbf{O}_t^{\text{sparse}}$
    $\mathcal{C} \leftarrow (\mathcal{G}_t, \mathcal{S}_t, \mathbf{R}_t)$
    $\widehat{\mathbf{O}}_t \leftarrow \mathbf{O}_t^{\text{full}}$
  else
    $(\mathcal{G}, \mathcal{S}, \mathbf{R}) \leftarrow \mathcal{C}$
    $\mathbf{O}_t^{\text{sparse}} \leftarrow \text{SparseAttn}(\mathbf{Q}_t, \mathbf{K}_t, \mathbf{V}_t; \mathcal{G}, \mathcal{S})$
    $\widehat{\mathbf{O}}_t \leftarrow \mathbf{O}_t^{\text{sparse}} + \mathbf{R}$
  end if
end for

Table 7: Dense warm-up and MC-Sparse anchor configurations. Warm-up entries denote count / total count. Hunyuan lists dual-stream and single-stream layers, respectively.

Model	Setting	Warm-up layers	Warm-up steps	Anchor steps
Wan2.1-1.3B-T2V	480p, 5s, text-to-video	1/30	10/50	$\{10, 26\}$
Wan2.1-14B-I2V	720p, 5s, image-to-video	1/40	10/50	$\{10, 26\}$
Wan2.1-14B-T2V	720p, 5s, text-to-video	1/40	10/50	$\{10, 26\}$
Hunyuan-13B	720p, 5.25s, text-to-video	1/20, 1/40	10/50	$\{10, 26\}$
Minimax-H3-Base	768p, 14.4s, text-to-audio-video	1/52	10/49	$\{10, 26\}$
HY3D-Internal	1536 resolution, image-to-geometry	1/40	1/12	$\{1\}$

C Experimental Details

C.1 Evaluation Protocol

Video generation evaluation. Following SVG-EAR, we use the Penguin Benchmark text-to-video prompts refined by the VBench team and the VBench image–prompt pairs with enhanced prompts for image-to-video generation. We randomly select 50 samples from each task's evaluation set. For Minimax-H3-Base, we further rewrite the test prompts with the official prompt-enhancement API and generate 345-frame videos, approximately 14.4 seconds long.

Analysis and ablation. We use Wan2.1-1.3B-T2V at 480p only for the analysis and ablation experiments. When computing the analysis metrics, we uniformly sample diffusion steps and transformer layers outside the dense warm-up regions. Analysis experiments use the same evaluation set as the corresponding generation task. We compute the reported metrics by uniformly sampling diffusion steps and transformer layers outside the dense warm-up regions.

Warm-up configuration. Table 7 lists the dense warm-up steps and layers for the video model configurations and the 3D generation experiment. Each entry gives the number of warm-up steps or layers divided by the corresponding total. For Hunyuan, $1/20$ and $1/40$ refer to the dual-stream and single-stream layers, respectively. MC-Sparse uses the same dense warm-up configuration as the corresponding baseline; its anchor steps are also listed in the table.

Baseline configuration. For Wan 2.1, we follow the baseline configurations used by SVG-EAR, with the key SVG2 and SVG-EAR parameters summarized in Table 8. Their warm-up schedules

Table 8: SVG2 and SVG-EAR settings following the SVG-EAR evaluation setup. Q_c and K_c denote the numbers of query and key clusters.

			TopP	
Model	Q_c	K_c	SVG2	SVG-EAR
Wan2.1-14B-T2V	300	1000	0.90	0.85
Wan2.1-14B-I2V	300	1000	0.90	0.85

Table 9: KV-selection granularity and query grouping at matched realized attention density s. *Align.* is the ratio of executed to nominal sparse interactions after tile alignment. Black rows include this inflation in the density budget, whereas gray rows ignore it[†].

Query grouping	KV granularity	Align.	Attention recall ↑			Output rel. L_1 ↓		
			s=0.1	s=0.2	s=0.3	s=0.1	s=0.2	s=0.3
No reorder	block	1.00	0.637	0.778	0.847	0.176	0.101	0.069
	token	1.00	0.745	0.857	0.910	0.130	0.070	0.044
k-means	block	2.08	0.743	0.830	0.874	0.160	0.098	0.071
	token	1.45	0.840	0.904	0.935	0.093	0.054	0.036
	block[†]	2.08	0.834	0.906	0.940	0.096	0.052	0.033
	token[†]	1.45	0.876	0.933	0.959	0.071	0.038	0.023
Fast PDDP (ours)	block	1.00	0.791	0.887	0.930	0.118	0.062	0.038
	token	1.00	**0.856**	**0.923**	**0.953**	**0.080**	**0.042**	**0.026**
Oracle (per query)	token	–	0.908	0.951	0.971	0.048	0.025	0.015

follow Table 7. For Sol-Attn, we use the diagonal estimator with $\tau = 0.52$, obtained by inverting the Gaussian approximation for a target attention density of 30%. We use the same Sol-Attn configuration for HY3D-Internal. For a fair comparison, none of the methods applies special treatment to condition tokens.

Efficiency measurement. We report speedup relative to dense attention based on DiT denoising latency, including dense warm-up computation. We do not report SpargeAttn's runtime because its implementation couples sparsification with low-precision attention, using INT8 or FP8 instructions for the QK and PV matrix multiplications, whereas the other methods use BF16, making the latencies not directly comparable. PISA's provided kernel cannot run efficiently on this model without modification. We therefore use the Sol-Attn kernel latency at the same attention density to estimate an upper bound on PISA's denoising speedup on HY3D-Internal.

C.2 Implementation Details

Operators and attention backend. Following the implementation practices of SVG2 and SVG-EAR, all Wan 2.1 and HunyuanVideo runs, including the dense-attention baseline, use optimized operators such as RoPE and normalization. Dense attention during warm-up uses FlashAttention-3 (FA3). Minimax-H3-Base uses the operators provided by Diffusers, with FA3 for both the dense-attention baseline and dense warm-up steps.

Hardware and parallelism. Wan 2.1, HunyuanVideo, and HY3D-Internal run on 1 Hopper GPU, including the condition encoder and VAE where applicable. Minimax-H3-Base runs on 8 Hopper GPUs: 1 handles condition encoding and VAE decoding, while the remaining 7 execute the denoiser with Ulysses sequence parallelism, which partitions the attention heads.

Two-Pass exact selection kernel. FlashAttention uses online softmax and does not expose the full normalized attention matrix needed for exact selection. We therefore use two passes: the first returns the dense output and row-wise log-sum-exp values; the second recomputes QK scores and uses these

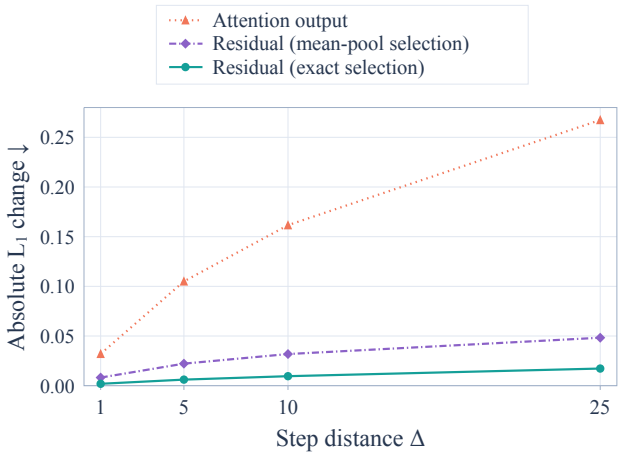

Figure 8: Absolute temporal change of attention outputs and dense–sparse residuals under token-level selection. Lower values indicate smaller absolute variation across Δ denoising steps.

Table 10: PSNR (dB) before and after adding our residual compensation at target attention density s. Δ is the PSNR gain from adding the residual. PISA retains its native block-statistics compensation in both configurations.

	$s = 0.20$		$s = 0.25$		$s = 0.30$	
Configuration	PSNR ↑	Δ	PSNR ↑	Δ	PSNR ↑	Δ
MC-Sparse w/o residual compensation	23.51	–	24.47	–	25.24	–
MC-Sparse	**27.05**	**+3.54**	**27.35**	**+2.88**	**27.54**	**+2.30**
Vanilla BSA	20.96	–	21.67	–	22.26	–
+ our residual compensation	22.21	+1.25	23.12	+1.45	23.90	+1.64
PISA	21.66	–	22.14	–	22.52	–
+ our residual compensation	22.15	+0.49	22.92	+0.78	23.42	+0.90

values to recover attention probabilities. We fuse group-wise aggregation into the second pass and apply top-K to the aggregated attention mass, avoiding materialization of the dense attention matrix.

Token-Sparse Attention kernel. We implement the token-sparse attention kernel in CuTeDSL with warp-specialized producer and consumer warps. Producer warps resolve selected indices and gather key and value rows asynchronously into shared memory. Consumer warpgroups apply WGMMA operations and online softmax, while a multi-stage pipeline overlaps the next gather with the current computation.

Table 11: Per-layer cost of offloading cached output residuals and selected KV indices to CPU memory. V is the transfer volume and C the sparse-attention FLOPs; C/V gives the break-even compute-to-PCIe-bandwidth ratio for overlap. Check marks indicate transfers hidden by prefetching one layer ahead in the measured setting.

Model	V (MB)	C (TFLOP)	C/V (FLOP/B)	Transfer (ms)	Compute (ms)	Transfer hidden
Wan2.1-14B-T2V	2435	29.1	11,409	47.7	208.8	✓
HunyuanVideo-13B	2710	34.6	12,164	52.9	244.5	✓

D MORE QUALITATIVE RESULTS

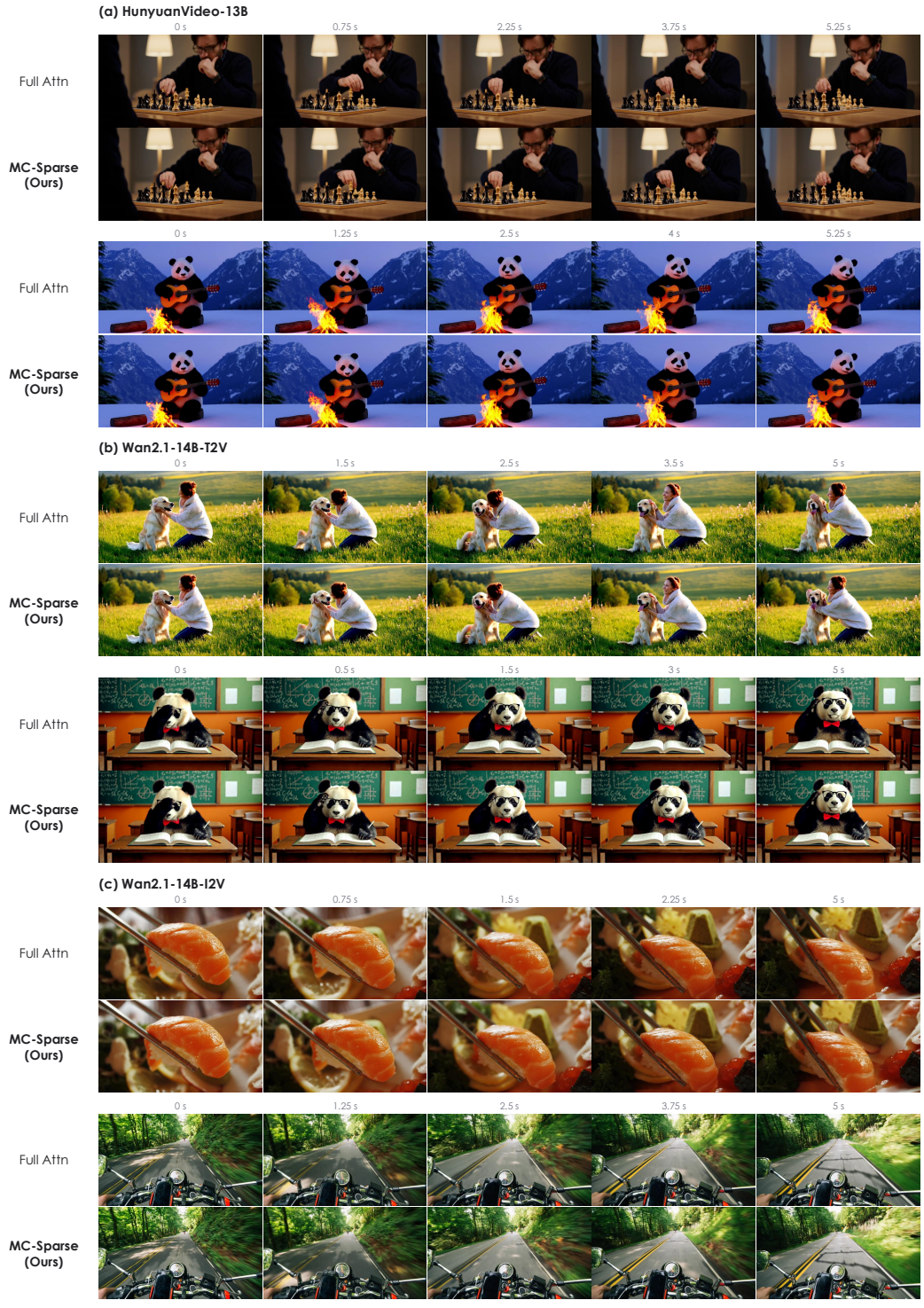

Figure 9: Qualitative comparison with full attention across video generation models.