Bidirectional Sparse Attention for Faster Video Diffusion Training

Chenlu Zhan* **Wen Li*** **Jun Zhang** **Chuyu Shen** **Hao Zhang†‡**
ByteDance

Abstract

Video diffusion Transformers (DiTs) produce high-quality videos, but the quadratic cost of full attention limits training on long spatiotemporal sequences. We propose Bidirectional Sparse Attention (BSA), a trainable attention framework that reduces computation on both the query and key–value sides. BSA skips locally redundant query computations, selects relevant key–value blocks using a per-query statistical threshold, and fills skipped attention outputs to preserve the original token layout. Custom Triton forward and backward kernels translate this structured sparsity into practical acceleration. At 153,600 tokens on Wan2.1-1.3B, BSA reduces forward attention-matmul FLOPs by approximately $20\times$ and achieves a reported attention-operator speedup of $17.79\times$. In a separate complete optimization-step benchmark on one H100-80GB GPU, it reduces step time from 10.40 s to 1.68 s ($6.20\times$), including selection, token filling, backward computation, and optimizer overhead. BSA maintains comparable overall generation quality: VBench-Total is 81.36 versus 81.43 for full attention, with small improvements in four reported submetrics. Additional experiments examine convergence, sparse-gradient correctness, token-filling error, and the role of sparsity annealing.

1 Introduction

Long-sequence modeling is a central computational challenge for video diffusion models [23, 27, 29]. High-resolution videos expand into tens or hundreds of thousands of latent tokens, making all-to-all attention expensive in both training and inference [14, 22, 30, 36]. Efficient attention must reduce this cost while preserving the spatiotemporal context needed for coherent generation.

Two forms of redundancy motivate our approach. First, adjacent video tokens often have similar query features and attention outputs, particularly in static backgrounds. Computing attention independently for every such query repeats similar work. Second, an individual query typically relies on only a subset of the available key–value (KV) context [20, 37, 41]. The useful subset varies with video content, layer, head, and diffusion timestep. Reducing only KV interactions leaves query-side redundancy unexploited; prescribing the same retained KV count for every query can allocate computation inefficiently.

We introduce **Bidirectional Sparse Attention (BSA)**, which jointly skips redundant query computations and dynamically selects KV blocks. Queries are partitioned into local spatiotemporal windows and compared with representative center tokens. Queries with distinctive features retain their own attention computation; skipped positions are filled using representative outputs. On the KV side, a threshold based on the empirical mean and standard deviation of block scores selects context separately for each retained query block. This preserves the dense output shape and token indexing expected by the surrounding DiT residual, normalization, and MLP paths.

*Equal contribution.
†Corresponding author.
‡Project Leader.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

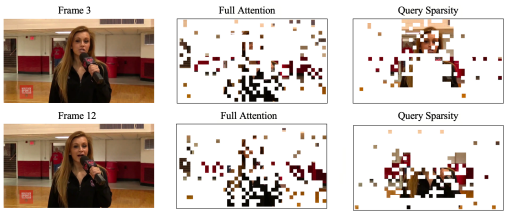

Figure 1: Query-side redundancy. Similar frames exhibit overlapping important-query regions under full attention. Query sparsification concentrates computation on locally distinctive content. The top-20% visualization is diagnostic and is not the query retention ratio used by the measured kernel.

The contribution is the joint design of these mechanisms and their trainable implementation, rather than query pruning in isolation. In contrast to physically merging the sequence, token filling approximates only the attention update: each original token retains its own residual path. Triton kernels support both forward and backward computation on the selected sparse interactions, and a gradual sparsity schedule allows the model to adapt during optimization.

Our evaluation separates three quantities that are often conflated: forward attention FLOPs, attention-operator latency, and complete optimization-step latency. At 153,600 tokens, the reported $17.79\times$ operator speedup is accompanied by a separately measured $6.20\times$ complete-step speedup. The latter reduces the timing-derived cost of a fixed 50,000-step budget from 144.4 to 23.3 GPU-hours on one H100. These costs do not measure time to a common loss threshold. Overall VBench quality remains close to full attention, while individual submetrics improve slightly; we do not claim universal quality improvement or exact equivalence to dense attention.

Our contributions are: (1) a bidirectional sparse operator combining query skipping, dynamic KV selection, and token filling; (2) trainable forward/backward kernels with explicit sparse-index mappings; and (3) an evaluation covering complete-step training efficiency, convergence, quality, and implementation correctness, including equal-sparsity controls and scheduling ablations.

2 Related Work

Sparse attention and token reduction. Sparse attention reduces long-context costs in language and multimodal models [1, 6, 8, 9, 13, 15, 17]. MoBA [20] and Native Sparse Attention (NSA) [37] demonstrate hardware-aware, trainable sparsity. Token merging and pruning also reduce redundant computation [2, 18, 21]; BSA shares their redundancy motivation. Its distinction is to skip selected query attention updates, restore the output layout through token filling, and combine this with per-query KV selection in a sparse forward/backward implementation. This leaves the surrounding dense-token DiT architecture intact, although attention outputs and gradients differ from full attention.

Efficient video diffusion. Video-specific sparse methods exploit spatial, temporal, or semantic structure [5, 16, 26, 28, 33, 35]. VSA [41] and VMoBA [32] provide trainable block-based alternatives. MoGA [12] uses a learnable semantic router to group tokens and integrates with existing attention kernels; it is also relevant to training-time acceleration. BSA instead separates query redundancy reduction from statistical KV selection and uses custom sparse kernels. Training-free compression and caching methods [4, 19, 39, 40] offer complementary acceleration paths. Our comparisons are specific to the evaluated configurations and do not establish a universal ranking of these approaches.

Linear attention and state-space models. The connection between Mamba and linear attention is analyzed by Han et al. [10]; CARE Transformer [42] targets efficient visual modeling through decoupled interactions. SANA-Video [3] uses block linear attention with constant-memory state, while LinGen [31] combines Mamba-based modeling with temporal attention. Long-context state-space video world models [24] extend temporal memory through state-space scanning and local attention. These methods change the underlying operator or architecture. BSA provides a complementary route by retaining softmax attention on selected interactions and the original DiT token layout.

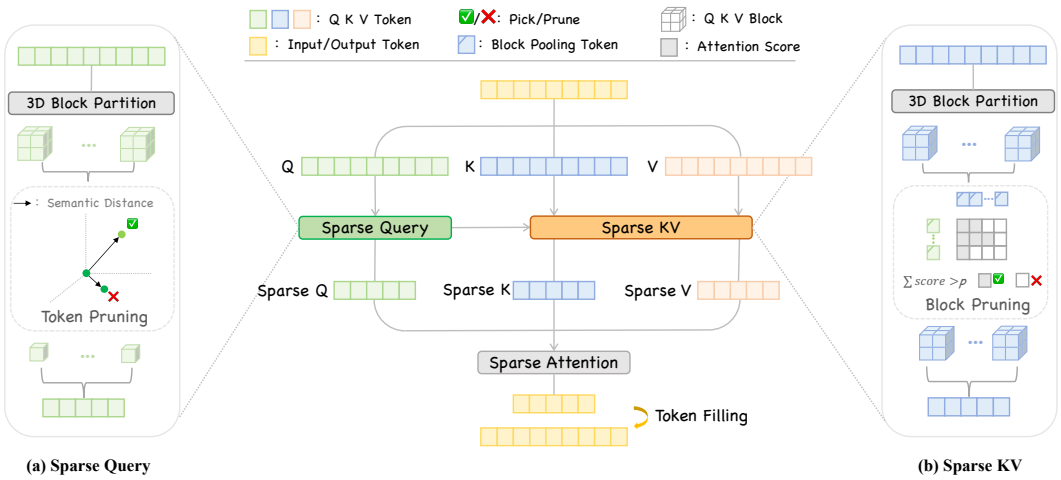

Figure 2: BSA pipeline. Spatiotemporal partitioning exposes local query redundancy. Selected queries attend to KV blocks that pass a per-query statistical threshold; token filling restores the full output layout. KV selection is a score-threshold test, not an iterative cumulative-probability rule.

3 Method

For a video latent of size $T \times H \times W$, let $N = THW$ be the sequence length and $Q, K, V \in \mathbb{R}^{N \times d}$ the projected features for one head. Dense attention computes

$$O = \mathrm{softmax}(QK^\top/\sqrt{d})V. \tag{1}$$

BSA approximates this operation through query selection (Sec. 3.1), dynamic KV selection (Sec. 3.2), and layout restoration. Section 3.3 describes the trainable implementation.

3.1 Query Sparsification and Token Filling

Spatiotemporal partitioning. We partition the latent into cuboids of size (C_t, C_h, C_w) and retain the mapping to original token indices. Within each cuboid, smaller windows of size (w_t, w_h, w_w) provide local representative tokens. This preserves spatial and temporal neighborhoods that may be obscured by a direct one-dimensional flattening. The default spatial cuboid is $(4, 4, 4)$ and the local window is $(2, 2, 2)$.

Query selection. Let $\mathcal{W}_b$ denote the tokens in a local window and c_b its center. We score non-center queries using their dissimilarity to the center,

$$a_i = 1 - \cos(q_i, q_{c_b}), \qquad i \in \mathcal{W}_b. \tag{2}$$

Distinctive queries with larger a_i retain separate attention computations. The representative must also be evaluated because its output is used for filling. Let $\mathcal{R}$ be the set of queries whose outputs are evaluated, including representatives used for filling. We distinguish the retained fraction from sparsity:

$$r_Q = \frac{|\mathcal{R}|}{N}, \qquad \text{query sparsity} = 1 - r_Q. \tag{3}$$

Selected queries are concatenated for sparse attention, and their original indices are kept for output restoration. The measured query-kernel operating point is $r_Q = 0.5$ with a query tile size $B_M = 64$; other retention ratios used in quality/FLOPs masking ablations do not imply separately implemented kernels.

Token filling. For retained query i, let o_i^s be its sparse attention output. The full-length update is restored as

$$\widehat{o}_i = \begin{cases} o_i^s, & i \in \mathcal{R}, \\ o_{c_b}^s, & i \in \mathcal{W}_b \setminus \mathcal{R}. \end{cases} \tag{4}$$

The residual path remains $x'_i = x_i + \widehat{o}_i$: skipped queries share an attention update but retain their individual input features and residual gradients. Gradients from filled positions accumulate into the representative output. Hard selection is treated as a discrete support decision; trainability refers to the differentiable computation on that support and through filling, not to differentiating the ranking operation. High cosine similarity motivates the approximation but alone does not guarantee small attention-output error. We therefore evaluate token-filling error directly (Sec. 4.4).

3.2 Dynamic KV Selection

For each retained query block i, let $S_i = (s_{i1}, \ldots, s_{in})$ contain its coarse inter-block scores over n candidate KV blocks. We use its empirical mean μ_i and standard deviation σ_i to define

$$p_i = \mu_i + \lambda\sigma_i, \qquad \lambda = U(1 - k/n), \qquad U(x) = \sqrt{\frac{x}{1-x}}, \quad 0 < x < 1. \tag{5}$$

Here k is a calibration parameter for the target retained KV-block count, so $r_{KV} = k/n$ and $\lambda = \sqrt{(n-k)/k}$. The retained support is

$$\mathcal{S}_i = \{j : s_{ij} > p_i\}. \tag{6}$$

The statistics adapt the cutoff to each score row without a global threshold, a learned gate, or sorting all KV scores. Larger λ yields a stricter cutoff for a fixed score row.

Statistical interpretation. Cantelli's inequality gives $\Pr(S - \mu \geq \lambda\sigma) \leq (1 + \lambda^2)^{-1}$ for a finite-variance scalar random variable and $\lambda > 0$. Setting this upper-bound expression to k/n motivates Eq. 5. This is a conservative calibration, not a Gaussian assumption, an exact-k selector, or a guarantee of optimal retained context. The number of selected KV blocks is determined by empirical scores; the bound does not guarantee preservation of dense softmax probability mass or generation quality. The dense warmup stage uses full support rather than applying the interior-domain calibration at its endpoint. A practical implementation must ensure nonempty support for degenerate score rows; the statistical calibration alone provides no such guarantee.

Sparse attention and cost. With $M_{ij} = 0$ for allowed token pairs and $-\infty$ otherwise, the selected-query output is

$$O^s = \mathrm{softmax}\Big(Q^s K^\top / \sqrt{d} + M\Big)\, V, \qquad \widehat{O} = \mathrm{Fill}(O^s). \tag{7}$$

Only $\widehat{O}$, after filling, has the original sequence length. Let $\mathcal{P}$ be the selected token pairs and $r = |\mathcal{P}|/N^2$ their retained fraction. Joint interaction sparsity is $1 - r$. The main long-sequence configuration has $r \simeq 0.05$, which is distinct from $r_Q = 0.5$. Forward attention matmuls cost approximately $4LhN^2dr$ FLOPs for L layers and h heads; Appendix E gives the block-count formula and its scope. Selection, filling, backward computation, and non-attention work are not part of this FLOPs estimate and are included in the complete-step timing instead.

Block-level FLOPs accounting. Let $N_Q = \lceil N/B_M \rceil$, $N_K = \lceil N/B_N \rceil$, and m_i count retained KV tiles for evaluated query tile $i \in \mathcal{I}_Q$. Counting a multiply-add as two FLOPs gives

$$F^{\mathrm{fwd}}_{\mathrm{BSA}} = 4LhB_M B_N d \sum_{i \in \mathcal{I}_Q} m_i, \qquad \frac{F^{\mathrm{fwd}}_{\mathrm{BSA}}}{F^{\mathrm{fwd}}_{\mathrm{dense}}} = \frac{\sum_{i \in \mathcal{I}_Q} m_i}{N_Q N_K} = r. \tag{8}$$

This form assumes full equal-capacity tiles and a common pattern count across heads and layers; otherwise their costs are summed separately. Partially filled tiles require active-length accounting. Query-token retention r_Q equals the query-tile retention fraction only when whole equal-sized tiles are selected; repacking must not count query sparsity twice.

3.3 Trainable Sparse Kernels

Our Triton implementation represents the sparse pattern with per-query lists `q2k_index` and counts `q2k_num`. Each query tile visits only its selected KV tiles, computes softmax attention, and accumulates the output. Padding masks handle partially occupied tiles. Query selection and token filling are connected through stored token indices.

Backward computation uses the same selected support. The query-outer traversal computes dQ; a lightweight inverse mapping from query-to-KV lists to KV-to-query lists enables a KV-outer traversal for dK and dV. Each KV tile accumulates contributions from its incident query tiles in a deterministic reduction, without `atomicAdd` for these attention-gradient accumulations. This mapping is part of the BSA timing. Appendix F details the traversal, while Sec. 4.4 compares outputs and gradients with a PyTorch implementation on identical sparse support. Such checks validate the sparse operator implementation, not equivalence to dense attention.

For row probabilities p and their gradient g, the score gradient is $dS = p \odot (g - \langle p, g \rangle)$. Inverse adjacency changes traversal, not normalization.

Differentiation through the restored layout. The distinction between a sparse attention update and a shortened token sequence is important during training. Let $g_i = \partial \mathcal{L} / \partial \widehat{o}_i$ be the gradient arriving at a restored position. For a representative c_b, token filling aggregates gradients from the positions assigned to it:

$$\frac{\partial \mathcal{L}}{\partial o^s_{c_b}} = g_{c_b} + \sum_{i \in \mathcal{W}_b \setminus \mathcal{R}} g_i. \tag{9}$$

A retained non-representative receives its own g_i. The residual connection separately propagates each position's gradient to its input. Thus skipping a query attention computation does not remove that position from the model or eliminate its residual learning signal. It does couple skipped attention updates through the representative, which is a source of approximation error. The reported sparse-reference checks use the same support decisions; they do not differentiate a counterfactual change in the selected indices.

Selection cost and computational scope. Given a row of n coarse KV scores, computing its moments and testing the threshold requires a linear scan in n, avoiding a full sort of that row. This statement concerns selection after scores are available; it does not make construction of all query–KV scores linear in sequence length. Query selection, index packing, inverse adjacency, and filling also consume time. The forward FLOPs estimate counts attention matrix multiplications on retained pairs, whereas the complete-step benchmark includes these auxiliary operations and the rest of the optimization loop. Separating these scopes is essential when evaluating a trainable sparse operator.

4 Experiments

4.1 Experimental Setup

Data and models. The original training study uses 300k videos from Vchitect T2V DataVerse [7], with shot segmentation, temporal truncation, and caption generation using Tarsier2 [38]. Wan2.1-1.3B is the principal backbone; the original study trains the 1.3B model from scratch on 8 H100 GPUs and fine-tunes Wan2.1-14B on 64 H100 GPUs. The original training budget is 30k steps. The additional training study extends the validation measurements to 50k steps and separately profiles complete optimization steps on one H100-80GB. We distinguish these protocols rather than treating every result as a single run. Further settings and cross-backbone results appear in Appendix A.

Baselines and metrics. We compare full attention with MoBA, VMoBA, VSA, and MoGA, and report both the original operating points and a separate control at 0.95 interaction sparsity. The controlled comparisons use the same data preprocessing, objective, backbone, training duration, validation prompts, evaluator, and fine-tuning budget. Only exposed sparsity parameters are tuned: VMoBA uses block sizes $\{32, 64\}$ and top-k $\{8, 16, 32\}$; MoBA uses top-k $\{16, 32, 64\}$; VSA varies its window and block-selection ratio from 0.1 to 0.3. The best stable settings are reported, with the optimizer and search budget held fixed (Appendix C). We report VBench [11] text consistency, background consistency, image quality, subject consistency, and VBench-Total where supplied. The aggregate score is not an average of the four displayed dimensions. Single-run quality numbers do not establish statistical significance.

Sparsity schedule. The implementation uses $r_Q = 0.5$ and anneals KV sparsity from denser attention. The original schedule increases the KV sparsity control by 0.03 every 30 steps up to 0.9, corresponding to an endpoint target $k/n = 0.1$. Because selection is threshold-based, this target is



Figure 3: Qualitative examples from the original evaluation. Corresponding frames illustrate facial details, scene structure, textures, and motion under full attention and BSA. These selected examples complement the quantitative metrics rather than establishing lossless generation.

not the realized retained fraction. The long-sequence experiment measures approximately 0.95 joint interaction sparsity. We additionally compare fixed-high sparsity, a 2k-step warmup followed by a jump, and gradual annealing to the same 0.95 joint-sparsity operating point.

4.2 Generation Quality and Attention Efficiency

Table 1: Original training-based results on Wan2.1-1.3B. Speedup is attention-operator speedup, not complete training-step speedup. Quality scores use the original percentage scale.

Tokens	Method	Sparsity	Text	BG	Image	Subject	Attn. speedup
23,296	Full attention	0	32.71	95.12	64.33	92.34	1.00×
	BSA	0.93	32.79	95.22	64.29	92.39	12.85×
153,600	Full attention	0	34.76	93.26	65.91	93.79	1.00×
	BSA	0.95	34.93	93.41	66.03	94.13	17.79×

Table 1 shows that the attention-level benefit grows with sequence length. At 23,296 tokens, estimated forward attention FLOPs decrease from 1.51×10^{12} to 1.05×10^{11}; at 153,600 tokens, they decrease from 6.99×10^{13} to 3.49×10^{12}, approximately a $20\times$ reduction. These figures cover attention matmuls and should not be interpreted as total training FLOPs. The 153,600-token generation evaluation gives VBench-Total 81.36 for BSA and 81.43 for full attention. Four reported submetrics increase slightly, but the aggregate does not; the supported claim is comparable quality at lower attention cost.

Why quality can be preserved. Training with the sparse operator allows the model to adapt to its deployed computation, avoiding the mismatch of an approximation introduced only at inference. Token filling preserves the surrounding token layout, and residual connections retain each token's input features. Removing redundant interactions could also have a regularizing effect, but this is an interpretation, not a demonstrated causal explanation for the small metric gains. The motion-heavy evaluation in Appendix G.6 shows that quality preservation is less reliable for rapid, fine-grained motion.

Matched-sparsity comparison. At 0.95 interaction sparsity, BSA reaches a better reported quality/attention-speed tradeoff than the tested MoBA, VMoBA, and VSA settings (Table 2). Thus the measured advantage is not explained solely by a larger skipped-interaction fraction. This comparison is specific to the selected implementations and tuning ranges; it does not remove all kernel or architecture differences.

Additional comparisons. The additional MoGA evaluation reports VBench-Total 81.10 and $11.6\times$ attention speedup, versus 81.36 and $17.79\times$ for BSA. Table 3 includes its submetrics; Table 4 separately compares reproduced SANA-Video and LinGen configurations. Additional original results cover Wan2.1-14B, the HunyuanVideo backbone, and training-free application. We keep their measurement scopes separate from the complete-step training results below.

Table 2: Separate equal-sparsity control at 153,600 tokens on Wan2.1-1.3B. All rows have 0.95 interaction sparsity. These are not the original operating points in Appendix C.

Method	VBench-Total	Attention speedup
MoBA	78.40	7.12×
VMoBA	78.90	6.41×
VSA	80.80	9.83×
BSA	81.36	17.79×

Table 3: Additional MoGA comparison in the evaluated long-video setting.

Method	Sparsity	Text	BG	Image	Subject	Total	Attn. speedup
Full attention	0	34.76	93.26	65.91	93.79	81.43	1.00×
MoGA	0.80	32.44	93.00	65.22	93.74	81.10	11.60×
BSA	0.95	34.93	93.41	66.03	94.13	81.36	17.79×

Architecture-changing alternatives. Table 4 reports reproductions at 153,600 tokens on H100 with common fine-tuning and evaluation budgets. SANA-Video and LinGen change the operator or architecture, so this is distinct from the same-backbone sparse controls. BSA has the highest reported aggregate quality and attention speedup among the three accelerated configurations. These results characterize the tested reproductions, not each method's published best performance. The MoGA comparison additionally reports 3.5× end-to-end speedup versus 6.2× for BSA; without a MoGA absolute-time breakdown, these ratios remain separate from the complete-step table below.

Table 4: Reported reproductions of architecture-changing alternatives, separate from same-backbone sparse controls.

Method	Text	BG	Image	Subject	Total	Attn. speedup
Full attention	34.76	93.26	65.91	93.79	81.43	1.00×
SANA-Video (reproduced)	34.05	93.01	65.32	93.59	79.62	8.40×
LinGen (reproduced)	34.55	93.17	65.62	93.42	80.05	10.60×
BSA	34.93	93.41	66.03	94.13	81.36	17.79×

4.3 Complete Training-Step Efficiency and Convergence

Timing protocol. We measure a complete optimization step on Wan2.1-1.3B at 153,600 tokens using one H100-80GB GPU and bf16. Global batch size and gradient accumulation are matched across methods; CUDA graphs are disabled. We report the median of 100 measured steps after 20 warmup steps, excluding first-step compilation. Timing includes data loading, forward, backward, the optimizer step, and non-attention overhead. BSA selection, token filling, and query-to-KV mapping inversion are included. Only the attention implementation changes.

Table 5: Complete optimization-step measurements. GPU-hours are derived from median step time for a fixed 50k-step budget on one GPU; they are not measured times to reach an identical validation loss.

Method	Step time (s)	Throughput (tokens/s)	Step speedup	GPU-hours
Full attention	10.40	14,769	1.00×	144.4
VMoBA	7.43	20,673	1.40×	103.2
VSA	3.35	45,851	3.10×	46.5
BSA	1.68	91,429	6.20×	23.3

BSA reduces median step time from 10.40 s to 1.68 s (Table 5). Throughput is the 153,600-token workload divided by step time. The 6.20× speedup is rounded from the reported timings. This is the direct evidence for faster training, distinct from the 17.79× attention-operator result. Non-attention work limits complete-step gains even when attention is highly sparse.

Validation trajectory. Table 6 reports validation loss at shared checkpoints. At 50k steps, the mean difference of 0.001 is smaller than the reported seed-level standard deviation of 0.002. Within this window, the losses remain close; this is neither a statistical equivalence test nor proof of identical asymptotic convergence. The original 30k-step loss plot remains in Appendix B.

Table 6: Validation loss at reported checkpoints (mean $\pm$ seed-level standard deviation).

Step	Full attention	BSA
20k	0.223 ± 0.004	0.222 ± 0.005
30k	0.215 ± 0.003	0.214 ± 0.004
40k	0.214 ± 0.002	0.213 ± 0.003
50k	0.213 ± 0.002	0.212 ± 0.002

The loss decreases between each pair of reported checkpoints for both methods. BSA's mean is lower by 0.001 at each checkpoint, while the supplied standard deviations overlap. The comparison therefore supports similar optimization behavior at these measured points, rather than a claim that sparsity improves convergence. Only the four reported checkpoints are available; they do not resolve behavior between checkpoints. Fixed-step GPU-hour estimates in Table 5 answer a separate resource question and do not establish a speedup to an identical target loss.

Forward/backward profiling. A separate attention-path benchmark at 153,600 tokens, bf16, head dimension 128, and 64×64 tiles reports 141.2/289.5 ms for dense forward/backward and 8.6/18.4 ms for BSA. Combined latency decreases from 430.7 to 27.0 ms ($15.9\times$). The BSA measurement includes support preparation, token filling, and mapping inversion as described in the added profiling record; it excludes the rest of the training loop. It must not be identified with either the original $17.79\times$ operator benchmark or the complete-step timing.

Table 7: Additional forward/backward attention-path latency; the rest of the optimization loop is excluded.

Stage	Full attention (ms)	BSA (ms)	Speedup
Forward	141.2	8.6	$16.4\times$
Backward	289.5	18.4	$15.7\times$
Combined	430.7	27.0	$15.9\times$

Dense-kernel control. The separate $1.07\times$ result over FlashAttention-3 uses dense all-to-all forward attention on one H100-80GB: Wan2.1-1.3B, 153,600 tokens, 24 heads, $d = 128$, and bf16. Median timing uses `triton.testing.do_bench` after warmup. The Triton path uses 64×64 tiles, autotuned 3/4/7-stage pipelines, and base-2 softmax, without unused causal/dropout branches. Dense mode still loads block indices. This modest gain reflects a shape-specific implementation comparison, not sparse acceleration or a general advantage over FA-3; no single cause is established without instruction-level profiling.

4.4 Sparse-Operator Correctness and Approximation Error

Reference computation. We compare the Triton implementation against a PyTorch sparse reference using identical selected query–KV interactions. This isolates numerical implementation consistency from the choice of support. The comparison covers both forward outputs and the dQ, dK, and dV gradients required for training. As shown in Table 8, output cosine similarity is 0.999999 and gradient cosine similarities range from 0.999990 to 0.999994. These observations support close directional agreement with the selected sparse computation. They are not a proof of exact arithmetic equality, and cosine similarity alone does not characterize absolute error.

Approximation against dense attention. A separate experiment compares token-filled outputs with dense-attention outputs at the implemented $r_Q = 0.5$, $B_M = 64$ operating point (Table 9). Relative L_2 error rises from 0.028 in shallow layers to 0.053 in deep layers, while cosine similarity

Table 8: Sparse-operator agreement with a PyTorch reference on identical selected support. Cosine similarity tests implementation consistency, not equivalence to dense attention.

Quantity	Forward output	dQ	dK	dV
Cosine similarity	0.999999	0.999990	0.999992	0.999994

remains above 0.995 for each reported group. The all-layer means are 0.040 and 0.9971, respectively. Preserving the token layout therefore does not imply identical activations: local representatives produce a small but measurable discrepancy in this setting, with greater error at depth.

Table 9: Layer-wise approximation error against dense attention.

Layer group	Relative L_2 error	Cosine similarity
Shallow (1–10)	0.028	0.9982
Middle (11–20)	0.041	0.9970
Deep (21–30)	0.053	0.9959
All-layer mean	0.040	0.9971

These two experiments address complementary questions. The sparse-reference test checks whether the kernel realizes the chosen operator; the dense comparison measures what is lost by approximating full attention. Neither replaces evaluation of training behavior and video quality. In particular, high feature similarity can coexist with errors in small moving objects or fine textures. We therefore interpret these diagnostics together with the extended validation losses, the schedule ablation, and the motion-heavy evaluation in Appendix G.6.

4.5 Ablations and Limitations

Sparsity schedule. Table 10 compares three ways of reaching joint sparsity 0.95. Fixed-high sparsity does not diverge in the tested setting, but yields worse loss and VBench-Total than gradual annealing. Warmup followed by an abrupt jump is intermediate. We therefore present annealing as an explicit stabilization recipe, rather than attributing its benefit to sparse support alone.

Table 10: Schedule ablation at the 0.95 joint-sparsity operating point. Full quality submetrics are in Appendix B.

Schedule	Validation loss	VBench-Total
Fixed sparsity 0.95 from step 0	0.221	80.47
2k warmup, then jump to 0.95	0.216	80.93
Gradual annealing to 0.95	0.212	81.36

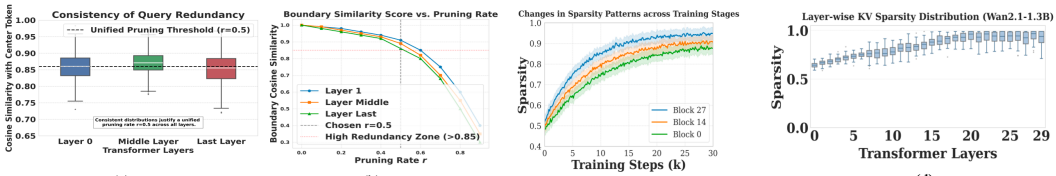

Figure 4: Original query-similarity and KV-sparsity diagnostics. Panels (a,b) describe query redundancy; (c,d) show variation across training stages and layers. Historical pruning-rate labels in the image denote the dropped fraction; our text uses retained ratio r_Q. At the implemented 0.5 operating point these values coincide.

Selection design. The component ablation (Table 11) gives $1.96\times$ attention speedup, increasing to $1.98\times$ with local windows. The statistical KV selector gives $6.12\times$ in its separate ablation, compared

with $6.05\times$ for a fixed cutoff. Appendix D reports the query-retention and KV-target sweeps and compares fixed-K, entropy-aware, and learned-gate alternatives. The statistical selector introduces no learned parameters but still has a retention target k and a schedule; it is not hyperparameter-free.

Table 11: Original component ablations at the short-sequence setting.

Configuration	Sparsity	Loss	Text	BG	Image	Subject	Speedup
Query only	0.50	0.211	32.83	95.25	64.34	92.44	$1.96\times$
Query + window	0.50	0.208	32.85	95.29	64.36	92.44	$1.98\times$
KV fixed cutoff	0.86	0.210	32.84	95.24	64.30	92.41	$6.05\times$
KV statistical cutoff	0.89	0.209	32.82	95.25	64.28	92.42	$6.12\times$
Full attention	0	0.213	32.71	95.12	64.33	92.34	$1.00\times$
Joint BSA	0.93	0.212	32.79	95.22	64.29	92.39	$12.85\times$

Alternative KV selectors. Table 12 compares the statistical threshold with fixed-K, entropy-aware selection, and a learned gate. Larger fixed-K improves quality but reduces acceleration in this sweep. The two-layer MLP gate adds 0.7M parameters and uses Gumbel-Softmax with a straight-through estimator. BSA gives the best reported quality/speed pair in this setting without additional learned parameters. This empirical comparison supports the chosen calibration; it does not make the statistical bound an optimality guarantee. A percentile rule is discussed in Appendix D without a quantitative result.

Table 12: Original threshold-design ablation at 153,600 tokens.

Selector	Extra parameters	VBench-Total	Attention speedup
Fixed-K, $K = 64$	0	80.21	$12.04\times$
Fixed-K, $K = 128$	0	80.78	$10.55\times$
Fixed-K, $K = 256$	0	81.05	$7.62\times$
Entropy-aware	0	80.94	$11.83\times$
Learned gate	0.7M	81.10	$12.41\times$
BSA statistical	0	81.36	$17.79\times$

Retention-target sensitivity. The KV calibration sweep in Appendix D favors $k = 32$ in the tested setting: measured sparsity is 0.890, TextConsis is 32.82, and attention FLOPs are 1.67×10^{11}. Reducing k to 4 raises sparsity to 0.986 but lowers TextConsis to 31.27; increasing k to 128 lowers sparsity to 0.560 and costs 6.68×10^{11} FLOPs. The selector therefore has a real quality/cost control, despite requiring no learned parameters. The separate query-only masking study gives VBench-Total 81.43/81.44/81.39 at $r_Q = 1.0/0.7/0.5$, with relative FLOPs 1.00/0.72/0.53. These are quality/FLOPs measurements, not kernel-speed measurements for every retention ratio.

Limitations. Center-token filling is less accurate for fast motion and fine textures. Threshold calibration controls selectivity but does not guarantee an exact support size, retained probability mass, or optimality. Reported speedups depend on sequence length, GPU, tiling, and timing scope. Evaluation spans a limited set of backbones and training windows, and most generation metrics lack uncertainty estimates. Further gains require addressing non-attention costs and improving representative selection in challenging motion regimes.

5 Conclusion

BSA combines query skipping, statistical KV selection, and token filling in a trainable sparse attention operator for video diffusion. On the evaluated long-sequence configuration, it reduces forward attention FLOPs by approximately $20\times$, achieves $17.79\times$ attention-operator speedup, and independently demonstrates $6.20\times$ faster complete optimization steps with comparable aggregate generation quality. The additional convergence, scheduling, approximation, and gradient checks clarify both the practical benefit and the limits of the approximation. Future work will study content-adaptive representatives and longer training horizons, with particular attention to fine motion, where local output sharing can discard information that remains important to the generated video.

Acknowledgments and Disclosure of Funding

All authors are affiliated with ByteDance.

References

[1] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[2] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. *arXiv preprint arXiv:2210.09461*, 2023. URL https://arxiv.org/abs/2210.09461.

[3] 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. *arXiv preprint arXiv:2509.24695*, 2025. URL https://arxiv.org/abs/2509.24695.

[4] Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. Delta-dit: A training-free acceleration method tailored for diffusion transformers. *arXiv preprint arXiv:2406.01125*, 2024.

[5] Hangliang Ding, Dacheng Li, Runlong Su, Peiyuan Zhang, Zhijie Deng, Ion Stoica, and Hao Zhang. Efficient-vdit: Efficient video diffusion transformers with attention tile. *arXiv preprint arXiv:2502.06155*, 2025.

[6] Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*, 2023.

[7] Weichen Fan, Chenyang Si, Junhao Song, Zhenyu Yang, Yinan He, Long Zhuo, Ziqi Huang, Ziyue Dong, Jingwen He, Dongwei Pan, et al. Vchitect-2.0: Parallel transformer for scaling up video diffusion models. *arXiv preprint arXiv:2501.08453*, 2025.

[8] Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, et al. Seerattention: Learning intrinsic sparse attention in your llms. *arXiv preprint arXiv:2410.13276*, 2024.

[9] Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. Longt5: Efficient text-to-text transformer for long sequences. *arXiv preprint arXiv:2112.07916*, 2021.

[10] Dongchen Han, Ziyi Wang, Zhuofan Xia, Yizeng Han, Yifan Pu, Chunjiang Ge, Jun Song, Shiji Song, Bo Zheng, and Gao Huang. Demystify mamba in vision: A linear attention perspective. *arXiv preprint arXiv:2405.16605*, 2024. URL https://arxiv.org/abs/2405.16605.

[11] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21807–21818, 2024.

[12] Weinan Jia, Yuning Lu, Mengqi Huang, Hualiang Wang, Binyuan Huang, Nan Chen, Mu Liu, Jidong Jiang, and Zhendong Mao. MoGA: Mixture-of-groups attention for end-to-end long video generation. *arXiv preprint arXiv:2510.18692*, 2025. URL https://arxiv.org/abs/2510.18692.

[13] Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024.

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

[15] Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. *arXiv preprint arXiv:2502.20766*, 2025.

[16] Xingyang Li, Muyang Li, Tianle Cai, Haocheng Xi, Shuo Yang, Yujun Lin, Lvmin Zhang, Songlin Yang, Jinbo Hu, Kelly Peng, et al. Radial attention: Sparse attention with energy decay for long video generation. *arXiv preprint arXiv:2506.19852*, 2025.

[17] Yucheng Li, Huiqiang Jiang, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Amir H Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, et al. Mminference: Accelerating pre-filling for long-context vlms via modality-aware permutation sparse attention. *arXiv preprint arXiv:2504.16083*, 2025.

[18] Junqing Lin, Xingyu Zheng, Pei Cheng, Bin Fu, Jingwei Sun, and Guangzhong Sun. Token pruning for in-context generation in diffusion transformers. *arXiv preprint arXiv:2602.01609*, 2026.

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

[20] Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.

[21] Wenbo Lu, Shaoyi Zheng, Yuxuan Xia, and Shengjie Wang. Toma: Token merge with attention for diffusion models. *arXiv preprint arXiv:2509.10918*, 2025.

[22] Guoqing Ma, Haoyang Huang, Kun Yan, Liangyu Chen, Nan Duan, Shengming Yin, Changyi Wan, Ranchen Ming, Xiaoniu Song, Xing Chen, et al. Step-video-t2v technical report: The practice, challenges, and future of video foundation model. *arXiv preprint arXiv:2502.10248*, 2025.

[23] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[24] Ryan Po, Yotam Nitzan, Richard Zhang, Berlin Chen, Tri Dao, Eli Shechtman, Gordon Wetzstein, and Xun Huang. Long-context state-space video world models. *arXiv preprint arXiv:2505.20171*, 2025. URL `https://arxiv.org/abs/2505.20171`.

[25] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

[26] Sucheng Ren, Qihang Yu, Ju He, Alan Yuille, and Liang-Chieh Chen. Grouping first, attending smartly: Training-free acceleration for diffusion transformers. *arXiv preprint arXiv:2505.14687*, 2025.

[27] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[28] Xin Tan, Yuetao Chen, Yimin Jiang, Xing Chen, Kun Yan, Nan Duan, Yibo Zhu, Daxin Jiang, and Hong Xu. Dsv: Exploiting dynamic sparsity to accelerate large-scale video dit training. *arXiv preprint arXiv:2502.07590*, 2025.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[30] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[31] Hongjie Wang, Chih-Yao Ma, Yen-Cheng Liu, Ji Hou, Tao Xu, Jialiang Wang, Felix Juefei-Xu, Yaqiao Luo, Peizhao Zhang, Tingbo Hou, Peter Vajda, Niraj K. Jha, and Xiaoliang Dai. LinGen: Towards high-resolution minute-length text-to-video generation with linear computational complexity. *arXiv preprint arXiv:2412.09856*, 2024. URL `https://arxiv.org/abs/2412.09856`.

[32] 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. *arXiv preprint arXiv:2506.23858*, 2025.

[33] Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, et al. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. *arXiv preprint arXiv:2502.01776*, 2025.

[34] Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring. *arXiv preprint arXiv:2503.16428*, 2025.

[35] Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, et al. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation. *arXiv preprint arXiv:2505.18875*, 2025.

[36] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

[37] Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025.

[38] Liping Yuan, Jiawei Wang, Haomiao Sun, Yuchen Zhang, and Yuan Lin. Tarsier2: Advancing large vision-language models from detailed video description to comprehensive video understanding. *arXiv preprint arXiv:2501.07888*, 2025.

[39] Zhihang Yuan, Hanling Zhang, Lu Pu, Xuefei Ning, Linfeng Zhang, Tianchen Zhao, Shengen Yan, Guohao Dai, and Yu Wang. Ditfastattn: Attention compression for diffusion transformer models. *Advances in Neural Information Processing Systems*, 37:1196–1219, 2024.

[40] Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattn: Accurate sparse attention accelerating any model inference. *arXiv preprint arXiv:2502.18137*, 2025.

[41] Peiyuan Zhang, Haofeng Huang, Yongqi Chen, Will Lin, Zhengzhong Liu, Ion Stoica, Eric Xing, and Hao Zhang. Vsa: Faster video diffusion with trainable sparse attention. *arXiv preprint arXiv:2505.13389*, 2025.

[42] Yuan Zhou, Qingshan Xu, Jiequan Cui, Junbao Zhou, Jing Zhang, Richang Hong, and Hanwang Zhang. CARE transformer: Mobile-friendly linear visual transformer via decoupled dual interaction. *arXiv preprint arXiv:2411.16170*, 2025. URL `https://arxiv.org/abs/2411.16170`.

A Experimental Details and Protocol Boundaries

The original Wan2.1-1.3B study uses 300k videos from Vchitect T2V DataVerse, processed by shot segmentation, temporal truncation, and Tarsier2 caption generation. The original setup uses 8 H100-80GB GPUs for the 1.3B model and 64 H100 GPUs for fine-tuning the 14B model. The additional complete-step timing is a separate single-H100 protocol. Fixed-budget GPU-hours derived from that timing must not be interpreted as the total cost of all experiments, preprocessing, tuning, or the original distributed training campaign.

The original settings are summarized in Table A1. Spatial partition size, original block capacity, and packed GPU tile size refer to different quantities. The rebuttal specifies the implemented query operating point as $r_Q = 0.5$, $B_M = 64$; its forward/backward microbenchmark uses $B_M = B_N = 64$, while the original configuration lists KV blocks of 128. These settings are reported separately rather than silently conflated.

Table A1: Original training hyperparameters and separately specified profiling configuration.

Setting	Value
Local window	$(2, 2, 2)$
Spatial cuboid	$(4, 4, 4)$
Query retention r_Q	0.5
Packed query tile B_M	64
Original KV tile	128
Additional profile tiles	64×64
Original training budget	30,000 steps
Extended validation window	50,000 steps
Optimizer	AdamW, $\beta = (0.9, 0.95)$
Learning rate / weight decay	10^{-4} / 0.01
Learning-rate schedule	Constant; 100 warmup steps
Original per-device batch size	1
Sequence lengths	23,296 / 153,600
Training objective	Flow matching
Timestep sampler	LogitNormal(0.0, 1.0)
Original dense attention implementation	FlashAttention-2
Dense specialization microbenchmark	FlashAttention-3 reference

The complete-step benchmark uses bf16, equal global batch size and gradient accumulation across methods, disabled CUDA graphs, 20 warmup steps, and the median of 100 measured steps. Compilation is excluded. Exact software versions and the numerical accumulation count are not included in the present description.

B Additional Training Diagnostics

B.1 Forward and backward timing

Table 7 reports the additional attention-path profile on one H100-80GB using bf16, 153,600 tokens, head dimension 128, and 64×64 tiles. Timing uses 20 warmup iterations followed by the median of 100 measurements and excludes compilation. The accompanying implementation description includes sparse selection, filling, and inverse mapping in the timed BSA path. It is distinct from the original operator-speed comparison and the full-loop measurements of Table 5.

B.2 Sparse-operator correctness

The Triton operator is compared with a PyTorch reference on *identical selected sparse support*. Table 8 in the main text reports the cosine-similarity checks. These test numerical implementation consistency, including gradients on the selected support. They do not test whether the selected support reproduces dense attention.

B.3 Token-filling approximation

Table 9 compares token-filled outputs with dense-attention outputs at $r_Q = 0.5$ and query tile size $B_M = 64$. The increasing error with depth illustrates why preserving tensor shape is not sufficient to guarantee preservation of dense activations. The reported mean relative L_2 error is 0.040; the sparse-reference gradient check above measures a separate property.

B.4 Convergence and scheduling

Table 6 reports the four supplied checkpoint summaries. The uncertainty values are seed-level standard deviations, not confidence intervals; no intermediate training trajectory is inferred from them. The number of seeds is not specified in the supplied record. The original loss plot (Figure A1) covers the original shorter training window and should not be used to imply a 50k-step per-step trajectory.

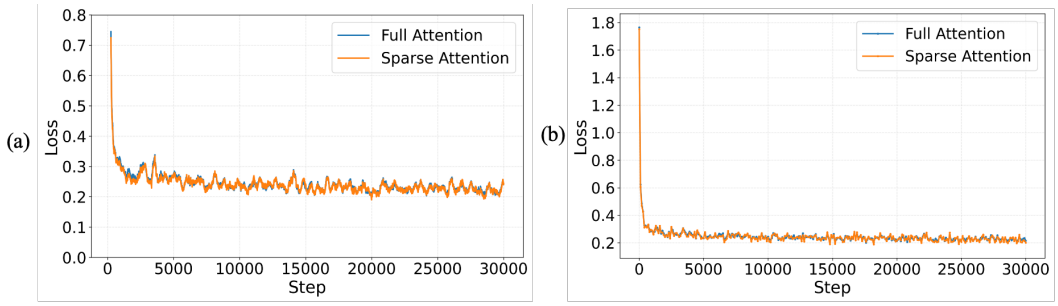

Figure A1: Original training and validation loss curves, retained from the submitted study. These are separate from the extended checkpoint summaries.

Table A2 gives all supplied quality dimensions for the three scheduling strategies. Fixed 0.95 sparsity did not diverge in this experiment, but gradual annealing gives the best loss and quality among the tested schedules. This observation does not establish stability for arbitrary backbones or sparsity levels.

Table A2: Detailed schedule ablation at joint sparsity 0.95.

Schedule	Val. loss	Text	BG	Image	Subject	Total
Fixed 0.95	0.221	33.80	93.07	65.01	93.14	80.47
2k warmup then jump	0.216	34.52	93.35	65.58	93.59	80.93
Gradual annealing	0.212	34.93	93.41	66.03	94.13	81.36

C Baseline Tuning and Additional Comparisons

C.1 Original operating points and matched sparsity

The additional controlled comparison matches objectives, preprocessing, backbone, training length, prompts, evaluator, and fine-tuning budget. Method-specific parameters are searched within stable ranges. Table A3 records the search ranges. Table A4 preserves the original operating-point quality and speed measurements. The equal-sparsity control is a separate experiment (Table 2), and its aggregate scores should not be combined with speedups from the original operating points.

Table A3: Reported baseline search ranges for the additional comparison.

Method	Parameters searched
VMoBA	Block size $\{32, 64\}$; top-k $\{8, 16, 32\}$
VSA	Window; block-selection ratio 0.1–0.3
MoBA	Top-k $\{16, 32, 64\}$
BSA	Query retention r_Q; annealed KV target k

Table A4: Original sparse-baseline operating points. Speedup is attention-level; these rows are not equal-sparsity controls.

Tokens	Method	Sparsity	Text	BG	Image	Subject	Speedup
23,296	MoBA	0.80	32.56	95.14	64.14	92.05	$1.20\times$
	VMoBA	0.18	31.53	95.17	64.15	92.36	$1.83\times$
	VSA	0.87	32.65	95.03	64.25	92.21	$4.50\times$
	BSA	0.93	32.79	95.22	64.29	92.39	$12.85\times$
153,600	MoBA	0.80	34.34	93.05	65.34	93.49	$2.30\times$
	VMoBA	0.19	32.55	93.31	66.01	94.06	$3.27\times$
	VSA	0.87	34.72	93.22	65.87	93.72	$6.20\times$
	BSA	0.95	34.93	93.41	66.03	94.13	$17.79\times$

Table A5: Quality dimensions reported for the separate 0.95-sparsity control. VBench-Total includes dimensions beyond those displayed.

Method	Text	BG	Image	Subject	VBench-Total
VMoBA	32.55	93.31	66.01	94.06	78.90
VSA	34.72	93.22	65.87	93.72	80.80
MoBA	34.34	93.05	65.34	93.49	78.40
BSA	34.93	93.41	66.03	94.13	81.36

C.2 MoGA and architecture-changing alternatives

MoGA [12] is a close trainable sparse-attention comparison with learnable semantic grouping. Table 3 records the added quality and attention-speed results at 153,600 tokens. The accompanying record reports end-to-end speedups of $3.5\times$ for MoGA and $6.2\times$ for BSA, but does not provide a MoGA absolute-time breakdown; these ratios are not used as replacements for the explicit complete-step measurements in Table 5. No MoGA FLOPs estimate is used in this comparison.

Table 4 presents the additional reproduced SANA-Video and LinGen comparison on the reported 153,600-token/H100 workload with common fine-tuning and VBench evaluation budgets. These methods change the architecture or operator; they are not identical-backbone attention replacements. The table is a configuration-specific reproduction and not a claim about each method's published best result.

D Additional Ablation Studies

D.1 Query and KV components

Table 11 preserves the original component ablations. A query-only configuration retains half the queries; adding local windows improves the recorded loss at nearly unchanged attention latency. The KV-only statistical threshold increases sparsity relative to a fixed cutoff. Joint sparsification provides the largest measured operator acceleration among these settings. Individual quality dimensions are mixed, so "lossless" is not an appropriate characterization of every row.

D.2 Query retention and dynamic strategies

In Table A6, r_Q denotes retention and query sparsity is $1 - r_Q$. The original FLOPs values increase with retention, confirming this interpretation of the table labels. These are query-side quality/FLOPs ablations, including masking simulations; they should not be read as measured kernels for every retention ratio. The quality trend is not monotonic in retention and therefore does not establish a universal retention optimum. The implemented operating point remains $r_Q = 0.5$.

The additional query ablation reports VBench-Total values 81.43, 81.44, and 81.39 at $r_Q = 1.0, 0.7, 0.5$, with relative FLOPs 1.00, 0.72, and 0.53. These quality/FLOPs measurements should not be combined with the differently scoped kernel timings above.

Table A6: Corrected query-retention labels for the original query-side ablation (23,296 tokens). FLOPs are reported attention estimates.

r_Q	$1 - r_Q$	Text	BG	Image	Subject	FLOPs
0.2	0.8	32.91	95.30	64.31	92.47	3.00×10^{11}
0.3	0.7	32.89	95.30	64.30	92.46	4.50×10^{11}
0.4	0.6	32.88	95.29	64.30	92.45	6.00×10^{11}
0.5	0.5	32.85	95.29	64.29	92.44	7.50×10^{11}
0.6	0.4	30.92	93.87	60.94	89.53	9.00×10^{11}
0.7	0.3	28.97	92.92	58.73	88.74	1.05×10^{12}

Table A7: Original dynamic query-strategy comparison at query sparsity 0.5.

Strategy	Validation loss	Text	BG	Image	Subject
Original	0.211	32.83	95.25	64.28	92.44
Spatial	0.207	32.87	95.28	64.32	92.47
Temporal	0.204	32.89	95.29	64.31	92.46
Spatial + temporal	0.202	32.91	95.30	64.33	92.48

D.3 KV calibration target

Table A8 sweeps the calibration target k, not an exact selected count. The reported sparsity and FLOPs vary with the realized support. The tested $k = 32$ setting offers a useful quality/cost compromise, but the smaller-k rows show measurable quality loss, so the selector is not insensitive to its target.

Table A8: Original KV-only target-count ablation.

k	Sparsity	Text	BG	Image	Subject	FLOPs
4	0.986	31.27	93.56	63.17	91.03	2.13×10^{10}
8	0.973	31.74	93.88	63.33	91.54	4.10×10^{10}
16	0.945	32.31	94.12	63.58	92.01	8.35×10^{10}
32	0.890	32.82	95.25	64.27	92.42	1.67×10^{11}
64	0.780	32.84	95.23	64.19	92.39	3.34×10^{11}
128	0.560	32.71	95.17	64.02	92.28	6.68×10^{11}

D.4 Alternative selection mechanisms

A fixed global cutoff is sensitive to score scale. Fixed-K adapts the selected identities but not their count. Percentile or cumulative-mass rules require additional selection work; learned gating adds parameters and training. Table 12 contains the supplied quantitative comparison for fixed-K, an entropy-aware rule, and a two-layer MLP gate trained with Gumbel-Softmax and a straight-through estimator. The percentile alternative is discussed qualitatively and has no numerical row here. The statistical rule has no learned parameters but does require k and a schedule.

E Attention FLOPs Accounting

Let L be the layer count, h the number of heads, d the head dimension, N the token count, and B_M, B_N the query and KV tile capacities. Set $N_Q = \lceil N/B_M \rceil$ and $N_K = \lceil N/B_N \rceil$. If $\mathcal{I}_Q$ contains the query tiles evaluated by the sparse kernel and m_i is the number of selected KV tiles for tile i, then, counting each multiply-add as two FLOPs,

$$F_{\mathrm{BSA}}^{\mathrm{fwd}} = 4LhB_M B_N d \sum_{i \in \mathcal{I}_Q} m_i, \qquad F_{\mathrm{dense}}^{\mathrm{fwd}} = 4LhN_Q N_K B_M B_N d \simeq 4LhN^2 d. \tag{10}$$

Each selected pair computes both $QK^\top$ and AV. Equation 10 assumes full equal-capacity tiles and a shared per-head/layer pattern count; when these counts differ, the sum is taken over layers and heads separately. For partially occupied query tiles, the exact token-pair count uses the active query and KV

lengths for each tile pair. Packing fewer retained queries into tiles must not be counted again as an independent sparsity factor.

For equal-capacity tiles, the interaction retained ratio is

$$r = \frac{\sum_{i \in \mathcal{I}_Q} m_i}{N_Q N_K}, \qquad \frac{F^{\text{fwd}}_{\text{BSA}}}{F^{\text{fwd}}_{\text{dense}}} = r, \qquad \text{joint sparsity} = 1 - r. \tag{11}$$

For token-level query selection, r_Q is the active query-token fraction. It equals $|\mathcal{I}_Q|/N_Q$ only when entire equal-sized query tiles are kept or dropped. This distinction matters when retained queries are repacked.

These expressions count forward attention matmuls only. They exclude softmax, projections, MLPs, normalization, score construction, support selection, token filling, mapping inversion, and backward operations. Query selection compares tokens with local representatives; a full sort within windows of size B costs $O(Nd + N \log B)$. Given a row of N_K coarse KV scores, thresholding costs $O(N_K)$; this is not a linear-in-sequence bound on constructing all query–KV block scores. Total selection overhead depends on the coarse scoring implementation and is best assessed through the full-path timings, rather than assumed negligible from FLOPs alone.

F Sparse Forward/Backward Implementation

Forward traversal. For each retained query tile, the kernel reads its list of selected KV tiles, forms scaled dot products, performs blockwise softmax accumulation, and writes the selected-query output. Token indices restore original ordering. The token-filling operation broadcasts representative outputs to skipped positions without changing the dense sequence expected by the residual and MLP paths.

Backward traversal and inverse adjacency. The forward adjacency is represented by `q2k_num` and `q2k_index`. For dQ, query tiles iterate over those same KV lists. For dK and dV, the implementation constructs the inverse adjacency `k2q`, so a KV tile iterates over every selected query tile that attended to it. Contributions are reduced within the KV-outer traversal and written once for that KV tile; the stated attention-gradient implementation avoids `atomicAdd`. Mapping construction is included in the reported BSA path timing. The no-atomics statement applies to dK/dV attention accumulation, not to an unspecified implementation of every auxiliary operation.

Gradient semantics. The hard support decision is held fixed for the local backward operation. Gradients propagate through selected query–KV interactions and through the token-fill mapping. Skipped positions contribute gradient to their representative attention output and preserve their own residual branch. Sparse-reference correctness must therefore be checked on the same support and fill mapping. It cannot demonstrate equality to dense gradients.

F.1 Dense specialization versus sparse acceleration

The original attribution experiment separates a BSA-pattern implementation using an FA-3 backend, a dense all-to-all Triton specialization, and the combined sparse system (Table A9). The dense $1.07\times$ figure is a forward-only microbenchmark using Wan2.1-1.3B, 153,600 tokens, 24 heads, $d = 128$, bf16, and one H100-80GB. It is not a gain from BSA sparsity.

Table A9: Original implementation-attribution results. These configurations are not additive or multiplicative causal components.

Configuration	Speedup vs. FA-3	VBench-Total
Dense FA-3	1.00×	81.43
BSA pattern + FA-3 backend	5.81×	81.18
Dense all-to-all BSA Triton	1.07×	81.43
BSA sparse pattern + Triton	17.79×	81.36

The dense specialization uses $B_M = B_N = 64$, autotuned 3/4/7-stage pipelines, base-2 softmax exponentiation, and no unnecessary causal/dropout branches. Dense mode still loads block indices.

A plausible explanation for its modest improvement is shape-specific tiling, pipelining, and specialization; without instruction-level profiling, we do not attribute it to one cause or claim a general advantage over FA-3. The larger BSA gain depends on skipping interactions and efficiently realizing the sparse pattern.

G Additional Generation Results

G.1 Training-free application

Table A10 reports the training-free quality scores. The separate component-level inference measurement below yields $3.64\times$ when all listed stages are included. Its scope differs from the complete optimization-step training result in Table 5.

Table A10: Original training-free quality evaluation on Wan2.1-1.3B.

Tokens	Method	Text	BG	Image	Subject
23,296	Full attention	30.51	93.13	61.99	89.43
	BSA	30.57	93.22	62.01	89.48
153,600	Full attention	31.98	93.41	62.97	90.31
	BSA	32.05	93.42	63.06	90.37

G.2 Inference component latency

Table A11 is the original 50-denoising-step inference breakdown (one H100-80GB, batch 1). Its original caption specifies fp16; the separate training-step profile uses bf16. The aggregate speedup is $536.8/147.3 = 3.64$. Text encoding is excluded and the listed VAE cost is included. This table is not a forward/backward training-step breakdown.

Table A11: Original inference component times with corrected total speedup.

Component	Full attention (s)	BSA (s)	Ratio
Self-attention (50 steps)	412.7	23.2	17.79×
FFN + convolution blocks	78.4	78.4	1.00×
Text cross-attention	21.6	21.6	1.00×
VAE encode/decode	14.3	14.3	1.00×
Normalization, projection, I/O	9.8	9.8	1.00×
Total	536.8	147.3	3.64×

G.3 Larger model

Wan2.1-14B is fine-tuned using the 300k-sample dataset on 64 H100 GPUs. Table A12 preserves the reported quality and attention speedups. We report measured attention acceleration separately from model-level FLOPs.

Table A12: Original Wan2.1-14B fine-tuning results.

Tokens	Method	Text	BG	Image	Subject	Attn. speedup
23,296	Full attention	54.98	96.42	85.12	95.67	1.00×
	BSA	56.33	97.04	86.32	96.74	13.69×
153,600	Full attention	57.31	96.98	86.43	96.35	1.00×
	BSA	58.12	97.05	87.13	97.08	18.93×

G.4 Cross-backbone evaluation

Table A13 preserves the original HunyuanVideo-family experiment, described in the submitted record as "HunyuanVideo-1.5B." The exact checkpoint identifier is not supplied; this experiment label

should not be read as establishing the parameter count of the public HunyuanVideo model [14]. A separate VBench-Long evaluation is listed in Table A14; its metrics use a different evaluation configuration and are not interchangeable with Table A13.

Table A13: Original HunyuanVideo-family quality evaluation; checkpoint label as reported in the submitted experiment record.

Tokens	Method	Text	BG	Image	Subject	Attn. speedup
23K	Full attention	29.98	93.11	61.97	90.26	$1.00\times$
	BSA	30.41	93.13	62.08	90.33	$13.52\times$
151K	Full attention	32.62	91.43	63.11	91.95	$1.00\times$
	BSA	33.11	91.53	64.15	92.01	$17.53\times$

Table A14: Separate original HunyuanVideo VBench-Long comparison.

Tokens	Method	Text	Subject	Image	Dynamic	Attn. speedup
23K	Full attention	26.32	94.80	67.41	56.20	$1.00\times$
	SVG	25.91	94.05	66.78	54.92	$2.41\times$
	BSA	26.18	94.62	67.20	55.84	$13.52\times$
151K	Full attention	26.10	94.55	67.12	67.84	$1.00\times$
	SVG	25.43	93.18	66.04	64.97	$2.58\times$
	BSA	25.97	94.21	66.83	67.05	$17.53\times$

G.5 Reference-based fidelity

Reference fidelity is measured against the dense model output using 200 VBench-Long prompts, Wan2.1-1.3B, 480×832 resolution, 81 frames, 50 sampling steps, and a fixed seed. Table A15 reports PSNR, SSIM, LPIPS, and the supplied Vision-Reward preference proxy. These are fidelity/quality measurements, separate from the 153K-token latency experiments. BSA is closer to the dense output than the listed sparse baselines under this protocol; finite PSNR and nonzero LPIPS explicitly show it is not identical to dense generation.

Table A15: Original dense-reference fidelity measurements.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	Vision-Reward ↑
Full attention	∞	1.000	0.000	0.62
SVG	24.9	0.842	0.131	0.55
SVG2	26.4	0.871	0.108	0.58
SpargeAttn	23.7	0.819	0.152	0.51
VSA	27.1	0.883	0.097	0.59
VMoBA	25.8	0.860	0.118	0.57
BSA	29.4	0.912	0.071	0.61

G.6 Motion-heavy clips

The original failure analysis partitions prompts by dense-reference Dynamic Degree: top-25% for motion-heavy clips and bottom-25% for static clips. Table A16 lists the directly reported dimensions. We compare the explicit dimensions rather than combining them into an additional aggregate. BSA's Dynamic Degree gap is $38.04 - 38.20 = -0.16$ on static clips and $80.48 - 81.55 = -1.07$ on motion-heavy clips. The latter is larger, supporting the stated motion limitation.

G.7 Additional sparse and caching baselines

Table A17 retains aggregate quality values from the original supplementary comparison. Attention compression and cross-step caching do not necessarily have the same acceleration scope. For this reason we do not present their heterogeneous source speedups as one uniform kernel benchmark. The VSA total of 81.04 here and its original $6.20\times$ speedup in Table A4 are from separate protocols, as clarified in the author response.

Table A16: Original motion-stratified VBench dimensions on Wan2.1-1.3B, 81 frames.

Split	Method	Text	Subject	Dynamic
Static	Full attention	26.13	95.42	38.20
	SVG	25.74	94.81	37.56
	VSA	25.91	95.05	37.92
	VMoBA	25.83	94.93	37.74
	BSA	26.05	95.28	38.04
Motion-heavy	Full attention	26.41	92.78	81.55
	SVG	25.07	90.42	76.18
	VSA	25.43	91.06	78.04
	VMoBA	25.21	90.78	76.99
	BSA	26.12	92.31	80.48

Table A17: Original supplementary aggregate-quality comparison; heterogeneous timing scopes are omitted.

Method	VBench-Total
Full attention	81.43
DiTFastAttn	79.86
XAttention [34]	80.21
TeaCache	80.05
Δ-DiT	79.74
SpargeAttn	80.42
SVG	80.71
SVG2	80.95
VMoBA	80.83
VSA	81.04
BSA (training-free)	81.21
BSA (trained)	81.36

G.8 Qualitative and original preference-study figures

Figure A2 retains additional qualitative examples. Figure A3 retains the original scaling, sparsity, and human-preference panels. The preference panel was described as comparing BSA and VSA on the 1.3B and 14B models using 210 randomly selected MovieGen-bench prompts [25]. Participant count, recruitment, instructions, compensation, and ethics-review details are not supplied in the current study record. Consequently, the main conclusions in this revision rely on the automated evaluations and measured timings, not on statistical claims from the preference panel.



Figure A2: Additional original qualitative comparisons across sequence lengths and resolutions.

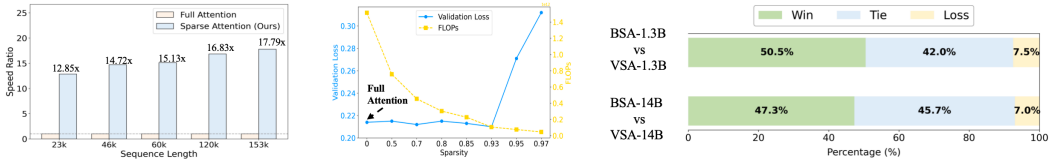

Figure A3: Original supplementary panels: (a) attention speedup versus sequence length; (b) validation loss and FLOPs versus sparsity; (c) the original human-preference study. Panel (c) is descriptive only; its participant and study-procedure details are not available in the current record.

H Limitations, Broader Impacts, and Reproducibility

Limitations. BSA relies on local representatives for skipped queries and coarse scores for KV selection. Rapid motion, fine textures, and unrepresentative center tokens can increase approximation error. The statistical threshold does not guarantee preservation of softmax mass and requires defined corner-case handling. The current experiment record lacks some implementation details, the seed count for the extended loss summaries, and participant details for the original preference study. Timing gains depend on hardware and workload, and no complete total-compute ledger is provided for all experiments. These limitations constrain generalization and exact reproduction.

Broader impacts. Reducing attention computation can lower the resource barrier for video-generation research. We have not measured energy or emissions savings directly. Faster generation can also facilitate misleading synthetic media, impersonation, and other misuse. Deployment should follow the underlying model and data licenses and appropriate provenance, consent, access-control, and content-safety practices. BSA is an attention algorithm and does not itself implement or validate these safeguards.

Code and data availability. The original submission checklist states that an anonymized code package accompanied the submission. The manuscript materials available for this revision contain LaTeX and figures, not that implementation package or a verified public repository. This version therefore does not claim a verified public code release. The algorithm, experimental summaries, and known protocol details are documented here; a public implementation, software manifest, and complete run configurations would improve reproducibility.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Sections 1 and 4.3 distinguish attention FLOPs, operator latency, and complete optimization-step latency. Quality claims are limited to the measured configurations.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 4.5 and Appendix H discuss approximation error, motion sensitivity, timing scope, incomplete protocol details, and limited uncertainty reporting.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Section 3.2 states the finite-variance condition and algebraic Cantelli calibration. Appendix E specifies assumptions for the FLOPs expressions; no guarantee of optimal context selection or dense-output equivalence is claimed.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [No]

   Justification: The method and recorded settings are described in Sections 3–4.1 and Appendix A, but the available record does not fully specify support corner cases, software versions, all baseline adaptation details, or the extended-study seed count.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The original submission described an anonymized supplementary code package. A public repository and its reproduction instructions have not been verified for this version; see Appendix H.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [No]

   Justification: The optimizer, original budget, data source, and measured timing procedures are reported, but some details needed for exact reproduction remain unspecified; see Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Table 6 reports seed-level standard deviations, but the seed count is unspecified. Most generation-quality results lack uncertainty estimates, and no statistical significance is claimed.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: GPU types/counts and complete-step timings are reported, including timing-derived fixed-budget GPU-hours. Total compute for all original runs, tuning, and preprocessing is not documented; see Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors state that the work follows the NeurIPS Code of Ethics. Potential misuse is discussed in Appendix H; the separate human-subject checklist items identify limitations in the study documentation.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix H discusses reduced computational barriers and risks of misleading synthetic media, along with deployment considerations.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This manuscript describes an attention algorithm and does not introduce a new dataset or release a new pretrained generative model. Deployment safeguards are discussed, not claimed as implemented, in Appendix H.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: Existing models, datasets, and baselines are credited, but this manuscript does not enumerate and verify all asset licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: The original record refers to supplementary code, but a complete public asset package, license, and associated documentation have not been verified for this version.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [No]

    Justification: The original 210-prompt preference-study panel is retained descriptively in Appendix G; participant instructions, compensation, and participant count are not supplied.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [No]

    Justification: The current record does not state participant risks, consent procedures, or approval/exemption status for the original preference study. No approval or exemption is inferred.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: BSA does not use an LLM as a core algorithmic component. Tarsier2 caption generation is identified in the data-preparation description; writing and formatting assistance is outside the core method.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.