WASP: WEIGHTED ASYMMETRIC SPARSE ATTENTION TO ACCELERATE BOTH PREFILLING AND DECODING IN LONG-CONTEXT MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context inference stresses attention in two different ways: prefilling is *compute*-bound, while decoding is bound by memory *bandwidth*. Sparse attention methods mitigate this problem, but usually deal with only one of the two bottlenecks. Many of them also rely on attention patterns that exist in specific architectures, on probes that overfit the question-at-the-end format of common benchmarks, or on rigid top-k budgets that fail to reflect how widely sparsity varies across layers, heads, tasks, and modalities. To this end, we introduce **W**eighted **A**symmetric **Sp**arse (WASP) attention, a training-free sparse attention designed to asymmetrically sparsify the two stages: block attention pools *queries* during prefilling and *keys* during decoding. Two primitives make WASP accurate: *inverse density weighted* (IDW) pooling, a TF–IDF-style summary that preserves the distinctive vectors that otherwise classical mean pooling would dilute, and an approximate Bernoulli sampler that adapts each head's budget to its own score distribution with no sorting. Across six text and multi-modal long-context benchmarks on Qwen3-8B, Llama-3.1-8B, and Qwen3-VL-8B, WASP matches the accuracy of dense attention on five of them while its attention layer runs $3.9\times$ faster at prefilling and $2.7\times$ at decoding, achieving an overall $2\times$ speedup that grows further with context length.

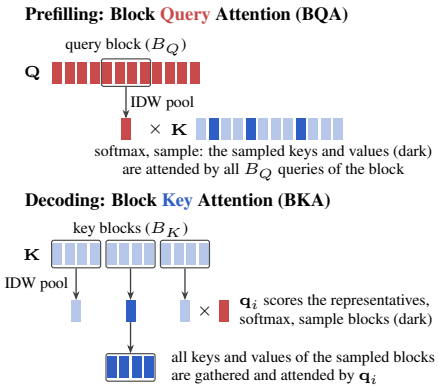
1 INTRODUCTION

LLMs increasingly work with long contexts. In agentic workflows the context accumulates over many turns, and a single agent attempt at a software engineering task processes 27M tokens on average (Desai et al., 2026). VLMs (Bai et al., 2025; Liu et al., 2023a; Lin et al., 2024; Wang et al., 2025b; Gemma Team et al., 2026) feed visual patches to the LLM backbone as separate tokens, and Qwen3-VL (Bai et al., 2025) turns an hour-long video into more than 200k tokens under its official evaluation settings. At these lengths, self-attention (Vaswani et al., 2017) with quadratic complexity dominates the cost of inference. However, most queries place nearly all of their attention mass on a small subset of the preceding tokens (Zhang et al., 2023), so self-attention is effectively sparse and can skip the rest with little change to its output and no training.

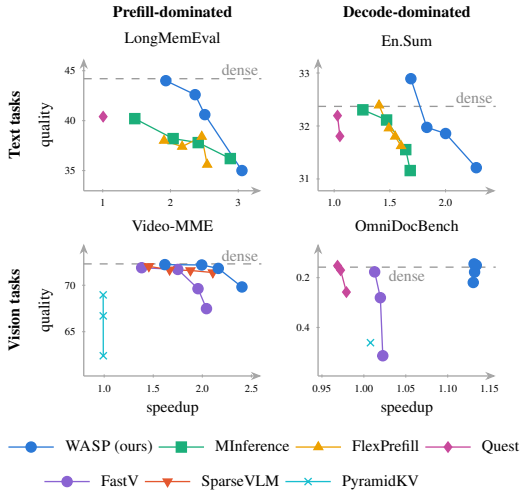
Training-free sparse attention methods exploit this sparsity in one of two ways. *Static eviction* methods discard key–value (KV) cache entries permanently (Li et al., 2024; Zhang et al., 2025b), although later context changes which entries matter. *Dynamic selection* methods keep the full cache and read a query-conditioned subset at each attention call (Tang et al., 2024). **W**eighted **A**symmetric **Sp**arse (WASP) attention belongs to this family and addresses five common limitations of prior work (Section 2). One line of dynamic methods relies on recurring structures in attention maps, which MInference (Jiang et al., 2024) and FlexPrefill (Lai et al., 2025) locate with a trailing-query probe and which XAttention (Xu et al., 2025) exploits through an estimator designed around them. However, these structures are architecture artifacts (Section 2), and the probes assume a question at the end of the prompt, which deployments do not guarantee (Section 2).

The other line uses no pattern prior and relies on *block attention*, which pools each block of queries or keys into a representative that decides whether the block is attended. Quest (Tang et al., 2024) pools per-dimension min–max bounds, whereas MoBA (Lu et al., 2025) and SpargeAttn (Zhang et al.,

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



(a) **WASP applies block attention asymmetrically, with BQA pooling *queries* to cut FLOPs at compute-bound prefilling and BKA pooling *keys* to cut HBM traffic at bandwidth-bound decoding.** One IDW representative scores the keys for a whole block of B_Q queries, which attend the sampled keys and values (dark). At decoding, $\mathbf{q}_i$ scores one representative per block of B_K cached keys, so each step reads only sampled blocks.



(b) **WASP's Pareto front lies above the baselines' at matched speed in all four panels.** Quality on Qwen3-8B and Qwen3-VL-8B is plotted against measured end-to-end speedup over dense attention (dashed line). Each method shows up to four configurations chosen from the full sweeps of Figs. 3 and 5, question-agnostic on LongMemEval and Video-MME. The OmniDocBench axis is reversed since it reports edit distance.

Figure 1: Overview of WASP, from its per-phase block attention (a) to its quality–speed Pareto fronts on four tasks (b).

2025a; 2026) apply the arithmetic mean, which dilutes a distinctive vector among its block-mates, and the choices have not been compared systematically (Section 2). Borrowing inverse document frequency from text retrieval, we propose *inverse density weighted* (IDW) pooling, which up-weights the vectors that differ from their block-mates and outperforms mean, min–max, and SVD pooling along the Pareto frontier (Section 5).

Neither line is designed around the two phases of inference. Prefilling is compute-bound, while decoding streams the KV cache from high-bandwidth memory (HBM) at every step and is bandwidth-bound (Dao, 2024; Zhong et al., 2024) (Section 2), yet both lines apply a single sparsification technique, which generally benefits only one of the two bottlenecks (Section A). WASP applies block attention asymmetrically (Fig. 1a), with block-query attention (BQA) pooling *queries* to cut FLOPs at prefilling and block-key attention (BKA) pooling *keys* to cut per-step HBM traffic at decoding.

Methods in both lines commonly pick the attended set with a fixed top-k (Tang et al., 2024; Jiang et al., 2024; Jones et al., 2026) or a sorting-based top-p (Lai et al., 2025), yet effective sparsity varies widely across layers, heads, tasks, and modalities (Section 2). WASP instead *samples* with a linearly approximated Bernoulli sampler, which adapts each head's budget to its own score distribution through one temperature τ and needs no sorting (Section 3.3).

Across six text and multi-modal long-context benchmarks on Qwen3-8B, Llama-3.1-8B, and Qwen3-VL-8B, WASP stays within about two points of dense accuracy on five of them at about 2× measured end-to-end speedup (Section 4.1), and on the prefill-dominated and decode-dominated tasks of Fig. 1b its Pareto front lies above the baselines' at matched speed. Under *question-agnostic prefilling*, where the context is encoded before the question arrives, as prefix caching requires, probe-based baselines lose up to 7 accuracy points while WASP moves by at most 2 points (Section 4.1).

Contributions. WASP is a training-free sparse attention that sparsifies a different operand in each phase, queries at prefilling and keys at decoding, and accelerates both at near-dense accuracy. **(1)** A block-attention framework instantiated per bottleneck, BQA for compute-bound prefilling and BKA for bandwidth-bound decoding, with load-balanced CUDA-graph-compatible Triton kernels

and the same graph-level engineering for every baseline (Section I), reaching about 2× measured end-to-end speedup at dense accuracy (Section 4.1). **(2)** IDW pooling, a drop-in representative for any block-attention method that beats mean pooling by up to 22 points, and a sorting-free Bernoulli sampler that beats rigid top-k by 15.2 (Section 5). **(3)** A question-agnostic prefilling protocol showing that without question foreknowledge probe-based methods lose up to 7 accuracy points and eviction up to 17, while WASP moves by at most 2 points (Section 4.1).

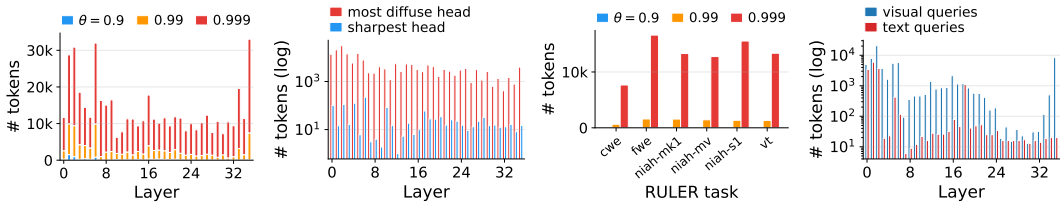
2 Limitations of Prior Sparse Attention

Limitation 1: Attention structures are architecture artifacts Many sparse attention methods are built on a predefined vocabulary of patterns in attention maps. The two structures at the core of these vocabularies are *vertical* lines, keys attended by most queries, and *slash* lines, attention at a fixed relative offset. MInference (Jiang et al., 2024) assigns each head one of three patterns offline (A-shape, vertical–slash, or block-sparse), and FlexPrefill (Lai et al., 2025) chooses per input between a vertical–slash pattern and query-aware block selection. XAttention's antidiagonal-sum pooling (Xu et al., 2025) is likewise designed around these structures. The origin of these structures is increasingly well understood. Slash lines arise from RoPE, whose medium and high frequencies concentrate attention at fixed relative offsets (Cheng et al., 2026; Yang et al., 2026; Barbero et al., 2025), while vertical lines track content salience such as sinks and the keys the current task retrieves. These structures are therefore architecture-contingent and do not generalize. They drift with the positional geometry across model families (MInference requires per-model offline calibration, and none exists for a new architecture), and they dissolve in multi-modal attention, whose spatio-temporal geometry produces none of the text-trained regularities. A selection mechanism that commits to a pattern vocabulary inherits the architecture these patterns were fitted on, and a general one should assume no patterns at all.

Limitation 2: Prefilling and decoding have different bottlenecks A standard self-attention head computes $\mathbf{O} = \mathrm{softmax}(\mathbf{Q}\mathbf{K}^\top/\sqrt{D})\mathbf{V}$ with $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{S\times D}$ for sequence length S and head dimension D, but the two inference phases execute it under opposite hardware bottlenecks. Prefilling processes queries in batched tiles, and a tile, e.g. 128, of queries streams the preceding KV cache only once while performing $\mathcal{O}(S^2D)$ calculation that is *compute*-bound. Decoding attends a single query per step, which shrinks the calculation to $\mathcal{O}(SD)$ but still streams the entire KV cache from HBM at every step, so it is *bandwidth*-bound. A roofline analysis (Fig. 8 in Section B) supports this intuition, placing prefilling's arithmetic intensity to the right of the ridge point and decoding's to the left. FLOPs scale with the number of queries while bytes scale with the cached keys and values, so a method that accelerates both phases cannot be symmetric. It should sparsify queries where compute dominates and keys where bandwidth dominates. Recent work reaches the same conclusion from other directions, with kernel-internal thresholds that skip the softmax at prefilling and the value loads at decoding (Yuan et al., 2026) and with a depth cutoff that shortens the prompt's path through the network while generated tokens keep the full depth (Oh et al., 2026).

Limitation 3: Block attention stands on its pooling function Block attention summarizes a block of vectors into one representative and lets a single score stand for B tokens, so both the expressiveness and the computational cost of the pooling function have to be considered when selecting it. Most methods default to the arithmetic mean (Lu et al., 2025; Zhang et al., 2025a; 2026), which dilutes distinctive vectors toward the block's common component, so the vectors a query would select the block for are averaged away. Our ablations show that our IDW dominates the mean and SVD pooling across the whole Pareto frontier and min–max (Tang et al., 2024) at every budget but the fastest (Section 5), and QuoKA (Jones et al., 2026) reports the same sensitivity, since scoring keys by the maximum over a chunk's retained queries beats their mean by 4 to 7 RULER points. Pooling and scoring run over the full sequence at every selection, so their own cost bounds the achievable speedup, as with XAttention's antidiagonal estimator, which costs $\Theta(S^2D/\mathrm{stride})$ and caps its end-to-end speedup below 1.9× at 100k even if the sparse attend were free (Section L).

Limitation 4: Sparsity varies across layers, heads, tasks, and modalities We judge how sparse attention is by measuring how many tokens are needed to cover the same fraction of attention mass under different conditions. For each query head we record the coverage budget k_θ, the smallest number of tokens whose attention mass, taken in descending order, reaches a fraction θ of the total (protocol in Section C). We measure it across layers, across heads within a single layer, across tasks



(a) Per layer. (b) Per head. (c) Per task. (d) Per modality.

Figure 2: **The sparsity of attention matrices is highly variable.** The number of tokens needed to cover the same fraction of attention mass varies across layers, attention heads, tasks, and modalities.

for the same model, and across modalities (Fig. 2). The budget varies significantly along every one of these axes, spanning over several orders of magnitude overall. Sparse attention therefore needs a dynamic budget, set at the level of individual heads, which is the requirement the sampling rule of Section 3.3 is built around.

Limitation 5: Probes overfit the benchmark format Most widely used long-context benchmarks place the question at the end of the prompt. Many sparse attention methods exploit this layout by using the trailing tokens of the prompt to probe what matters, among them MInference and FlexPrefill, which estimate pattern positions from the last 64 or 128 queries, SnapKV and PyramidKV, which score the cache against a trailing observation window (Li et al., 2024; Cai et al., 2025), and SparseVLM, which rates visual tokens with the text that follows them (Zhang et al., 2025b). Actual deployments do not follow this layout, since prefix caching may prefill the context before any question exists and multi-turn agents end their prompts with tool outputs that arrived last. Our experiments show that withholding the question from the context prefilling, as prefix caching does, costs these methods up to 17 accuracy points (Section 4.1), so selection should not depend on the question position.

3 WASP: Asymmetric Sparse Attention

We propose WASP to address the five limitations of Section 2 with a block attention design built on the following choices. **(1) Block attention without pattern priors.** Selection scores pooled block representatives and assumes no attention pattern, so the same mechanism serves text and vision. **(2) Asymmetry.** Block attention pools queries during prefilling and keys during decoding, so each phase sparsifies against its own bottleneck (Section 3.1). **(3) IDW pooling.** The representatives are computed by inverse density weighting, which keeps a block's distinctive vectors from being averaged away at under 1% of the prefilling cost (Section 3.2). **(4) Sampling with a dynamic budget.** The attended set is sampled from the block scores rather than cut by top-k or top-p, so each head keeps a budget of its own without sorting (Section 3.3). **(5) No probes.** Every selection is made by the queries themselves, per block at prefilling and per step at decoding, so it does not depend on where the question sits.

3.1 Asymmetric block attention

During prefilling, we approximate self-attention with BQA. We first pool $\mathbf{Q}$ to representatives, one for every B_Q adjacent queries without overlap (default $B_Q = 1024$), so that $\mathrm{Pool}(\mathbf{Q}) \in \mathbb{R}^{\lceil S/B_Q \rceil \times D}$. We then compute $\mathrm{Pool}(\mathbf{Q})\mathbf{K}^\top$ to give each key a sampling probability, sample the keys and values for each query block accordingly, and compute self-attention over the sampled keys and values,

$$\mathbf{O}_{\mathrm{BQA}} = \mathrm{softmax}\Big(\frac{\mathbf{Q}\mathbf{K}_{\mathrm{pre}}^\top}{\sqrt{D}}\Big)\mathbf{V}_{\mathrm{pre}}, \quad (\mathbf{K}_{\mathrm{pre}}, \mathbf{V}_{\mathrm{pre}}) = \mathrm{Sample}\Big(\mathbf{K}, \mathbf{V}; \mathrm{softmax}\Big(\frac{\mathrm{Pool}(\mathbf{Q})\,\mathbf{K}^\top}{\tau\sqrt{D}}\Big)\Big),$$

where $\mathbf{K}_{\mathrm{pre}}, \mathbf{V}_{\mathrm{pre}}$ are the keys and values sampled during prefilling and τ is the temperature that controls the budget. Computing $\mathrm{Pool}(\mathbf{Q})\mathbf{K}^\top$ costs B_Q times fewer FLOPs than $\mathbf{Q}\mathbf{K}^\top$ in dense self-attention, and $\mathbf{K}_{\mathrm{pre}}$ is far smaller than $\mathbf{K}$, so the attention computation shrinks accordingly.

During decoding, instead, we approximate self-attention with BKA, which reuses the above operators over keys. Pool maps $\mathbf{K}$ to one representative for every B_K adjacent keys (default $B_K = 16$),

computed once at the end of prefilling, and at each step the query scores the representatives, Sample draws whole key blocks with their values, and the query attends over them,

$$\mathbf{O}_{\mathrm{BKA}} = \mathrm{softmax}\Big(\frac{\mathbf{Q}\mathbf{K}_{\mathrm{dec}}^{\top}}{\sqrt{D}}\Big)\mathbf{V}_{\mathrm{dec}}, \quad (\mathbf{K}_{\mathrm{dec}}, \mathbf{V}_{\mathrm{dec}}) = \mathrm{Sample}\Big(\mathbf{K}, \mathbf{V}; \mathrm{softmax}\Big(\frac{\mathbf{Q}\,\mathrm{Pool}(\mathbf{K})^{\top}}{\tau\sqrt{D}}\Big)\Big).$$

The two definitions share the same form and differ only in the highlighted operand, $\mathrm{Pool}(\mathbf{Q})$ at prefilling against $\mathrm{Pool}(\mathbf{K})$ at decoding, which is what makes each phase sparsify against its own bottleneck. Scoring the $\sim S/B_K$ representatives reads B_K times fewer keys than scoring $\mathbf{K}$, and the attend reads only the sampled blocks, so the per-step memory traffic shrinks accordingly. In practice, both phases add the sink tokens and a recent window to the sampled keys and values, and the computation follows token order so that causal attention is preserved. For visual tokens, we reorder the KV cache at write time so that each key block covers a spatially contiguous region (Section F). The description uses a single head for brevity, and under grouped-query attention every query head pools, scores, and samples independently over the keys and values of its group, at prefilling and at decoding alike.

3.2 Inverse density weighted pooling

We propose IDW pooling to instantiate $\mathrm{Pool}(\cdot)$ function. Pool slices its input matrix into blocks of B consecutive rows, where B is the block size, $B{=}B_Q$ for queries and $B{=}B_K$ for keys, and summarizes each block into one vector independently,

$$\mathrm{Pool}(\mathbf{X}) = \big[\mathrm{IDW}(\mathbf{X}'_1); \mathrm{IDW}(\mathbf{X}'_2); \ldots; \mathrm{IDW}(\mathbf{X}'_{\lceil S/B \rceil})\big], \qquad \mathbf{X}'_j = \mathbf{X}_{(j-1)B+1:\,jB},$$

where $\mathbf{X}'_j$ is the jth block of $\mathbf{X} \in \mathbb{R}^{S\times D}$, carries vectors from index $(j-1)B+1$ to index jB. The last block may contain fewer than B vectors when S is not divisible by B. IDW is inspired by TF–IDF in text retrieval, which scales a term's frequency by its inverse document frequency (Sparck Jones, 1972), and likewise weights each vector by how common it is within its block. Given a block $\mathbf{X}'_j$, write $\mathbf{x}'_1, \ldots, \mathbf{x}'_B$ for its rows, indexed within the block. We define the *density* of a row as its total similarity to the rows in its block, measured by the dot-product similarity,

$$d_i = \sum\nolimits_{k=1}^{B} \mathbf{x}'_i \cdot \mathbf{x}'_k,$$

and IDW then pools the block by weighting each row with the inverse of its density,

$$\mathrm{IDW}(\mathbf{X}'_j) = \Big(\sum\nolimits_{i=1}^{B} w_i\,\mathbf{x}'_i\Big) \Big/ \Big(\sum\nolimits_{i=1}^{B} w_i\Big), \qquad w_i = \frac{1}{\max(d_i, \varepsilon)},$$

where $\varepsilon{=}10^{-6}$ guards the division. In this way, common vectors within a block who has high density will be assigned with a low weight, and distinctive vectors will have a high weight in the block representative.

3.3 Distribution-aware sampling

To avoid sorting, neither phase uses top-k or top-p selection. The softmax of Section 3.1, taken over $\mathrm{Pool}(\mathbf{Q})\mathbf{K}^{\top}$ at BQA and over $\mathbf{Q}\,\mathrm{Pool}(\mathbf{K})^{\top}$ at BKA, assigns each candidate j, a key at BQA and a key block at BKA, a probability p_j, and we sample the kept set from this distribution directly.

The simplest sampler draws i times from the distribution, once per position preceding a query at position i, and keeps every candidate drawn at least once, so that candidate j is included with probability $1-(1-p_j)^i$, the inclusion probability of probability-proportional-to-size sampling (Tillé, 2006). Since p_j is of the order of $1/i$, this expression cancels catastrophically unless it is evaluated in high precision, and the exact form also loses accuracy in practice (Section D). We therefore use its first-order approximation $i\,p_j$ together with a uniform floor U/i, and keep candidate j independently with probability

$$p_j^{\mathrm{keep}} = \min\Big(1,\ \max\Big(i\,p_j,\ \frac{U}{i}\Big)\Big).$$

A candidate whose probability exceeds the uniform value $1/i$ is kept with certainty, the others are kept in proportion to their probability, and the floor adds an expected U positions drawn uniformly at

random whatever the scores say, which covers keys that a pooled representative under-scores. One random number per candidate suffices.

At BQA the sampled units are individual keys. One sampled set serves the whole query block and i is the position of the first query of the block, so that the set is causal for every query in it, while attention within the block is computed densely. At BKA the sampled units are key blocks of B_K keys, i is the position of the current query, and a sampled block enters the attention with all of its keys, so that the floor contributes an expected U/B_K blocks. We use $U{=}1024$ in both phases.

The kept set has no fixed size. A head whose probability mass concentrates on few candidates keeps few and a head whose mass is spread keeps many, without per-layer or per-head calibration. The temperature τ of Section 3.1 divides the scores before the softmax, so a larger τ flattens the distribution and keeps more candidates while a smaller τ sharpens it and keeps fewer.

4 Experiments

We evaluate WASP on a single H200 in bf16 with greedy decoding, in two modalities, text and video understanding, and in each modality on prefill-dominated and on decode-heavy tasks. For text we use Qwen3-8B (Yang et al., 2025) and Llama-3.1-8B (Grattafiori et al., 2024), which share every WASP setting except the sampling temperature τ. The prefill-dominated benchmarks are RULER (Hsieh et al., 2024) at 100k context, a synthetic retrieval suite, and LongMemEval-S (Wu et al., 2025) at 102–112k, natural conversational memory that is closer to actual use, and the decode-heavy task is ∞Bench En.Sum (Zhang et al., 2024), book summarization with 126k prefilling and about 900 decoding, so that decoding takes more than 40% of dense end-to-end time. For video understanding we use Qwen3-VL-8B (Bai et al., 2025) with the same hyper-parameters as Qwen3-8B, on Video-MME (Fu et al., 2025) and LVBench (Wang et al., 2025a), prefill-dominated video QA with up to about 220k visual tokens, and on OmniDocBench (Ouyang et al., 2025), decode-heavy document OCR with about 4.6k prefilling and 1.1k decoding, which also probes the short-context regime where attention savings shrink.

The text baselines are the widely used training-free methods MInference (Jiang et al., 2024) and FlexPrefill (Lai et al., 2025), which sparsify prefilling only, and Quest (Tang et al., 2024), which sparsifies decoding only, and the vision baselines are the token-reduction methods FastV (Chen et al., 2024), SparseVLM (Zhang et al., 2025b), and PyramidKV (Cai et al., 2025). Each baseline runs at the configuration its paper recommends with only its budget knob swept, on identical prompts, with the same optimizations as WASP, and normalized to a FlashAttention-2 dense reference (Sections I and J). XAttention (Xu et al., 2025), whose antidiagonal pooling caps its end-to-end speedup below the range WASP operates in, is reported in Section L. Every method is evaluated under two protocols, the standard one, where the question is part of the one-shot prefilling, and question-agnostic prefilling, where the context is encoded before the question exists, as with prefix caching, and where for video the frames are encoded and compressed before the question and instruction arrive (Section H), the protocol closer to actual deployment. Speed comes from a matched benchmark on identical prompts, and end-to-end speedup composes the measured prefilling and decoding rates with each benchmark's dense generation length (Section J).

4.1 Text tasks

Table 1 reports both protocols and Fig. 3 shows the question-agnostic sweeps. Withholding the question moves the baselines but not WASP: the probe-based MInference and FlexPrefill lose up to 7 points on RULER and 6 on LongMemEval across their sweeps (Table 12), and on Llama-3.1-8B every entry that sits above dense under the standard protocol falls below it, whereas WASP selects with the true query at the moment of use and moves by at most 2 points.

On the prefill-dominated tasks, read under question-agnostic prefilling, the two benchmarks rank the methods differently. On the synthetic RULER, MInference keeps the most accuracy on Qwen3-8B, since its vertical–slash prior fits the needle layout, and it stays 3 to 7 points above WASP beyond $1.9\times$, while WASP leads FlexPrefill along the whole frontier. On the natural LongMemEval, WASP is the best method at every speed, within 0.2 points of dense at $1.9\times$ and 3 to 6 points above the best baseline point across the frontier. On Llama-3.1-8B the ranking is the same: at the MInference speed ($\tau{=}3.0$) WASP leads on RULER and sits within 0.4 points of FlexPrefill on LongMemEval, with

Table 1: **WASP is the fastest method on every benchmark and model with matched accuracy, and its accuracy is invariant to the protocol, whereas the probe-based baselines lose accuracy once the question is withheld.** Results on the text benchmarks under the standard protocol (*std*) and question-agnostic prefilling (*q-agn.*), with measured end-to-end speedup. Arrows represent the score change from *std* to *q-agn.*, and WASP changes by at most 1 point owing to its probe-free design. Bold marks the best non-dense entry per column and model. We provide full details in Section G.

Model	Method	RULER			LongMemEval			En.Sum	
		std	q-agn.	spd	std	q-agn.	spd	R-L	spd
Qwen3-8B	Dense (FA2)	72.0	72.0	1.00×	44.2	44.2	1.00×	32.4	1.00×
	MInference s=6096	**72.0**	**70.1** ↓1.9	1.31×	39.8	40.2 ↑0.4	1.47×	32.3	1.26×
	FlexPrefill γ=0.95	70.6	67.4 ↓3.2	1.66×	44.0	38.0 ↓6.0	1.90×	32.4	1.40×
	Quest b=2048	67.1	67.1	1.00×	40.0	40.4 ↑0.4	1.01×	31.6	1.05×
	WASP τ=4.0	69.8	70.0 ↑0.2	1.70×	**44.2**	**44.0** ↓0.2	1.82×	**32.6**	1.52×
	WASP τ=2.5	63.6	64.1 ↑0.5	**2.14**×	42.0	42.6 ↑0.6	**2.36**×	32.0	**1.83**×
Llama-3.1-8B	Dense (FA2)	75.6	75.6	1.00×	36.4	36.4	1.00×	31.6	1.00×
	MInference s=6096	**76.2**	74.4 ↓1.8	1.70×	37.6	32.6 ↓5.0	1.68×	30.7	1.37×
	FlexPrefill γ=0.95	74.4	72.3 ↓2.1	1.89×	**38.4**	**35.6** ↓2.8	1.86×	30.9	1.40×
	WASP τ=3.0	74.4	**74.7** ↑0.3	1.73×	35.0	35.2 ↑0.2	1.72×	**31.7**	1.59×
	WASP τ=2.0	73.3	73.8 ↑0.5	**2.11**×	36.0	35.0 ↓1.0	**2.09**×	30.6	**1.86**×

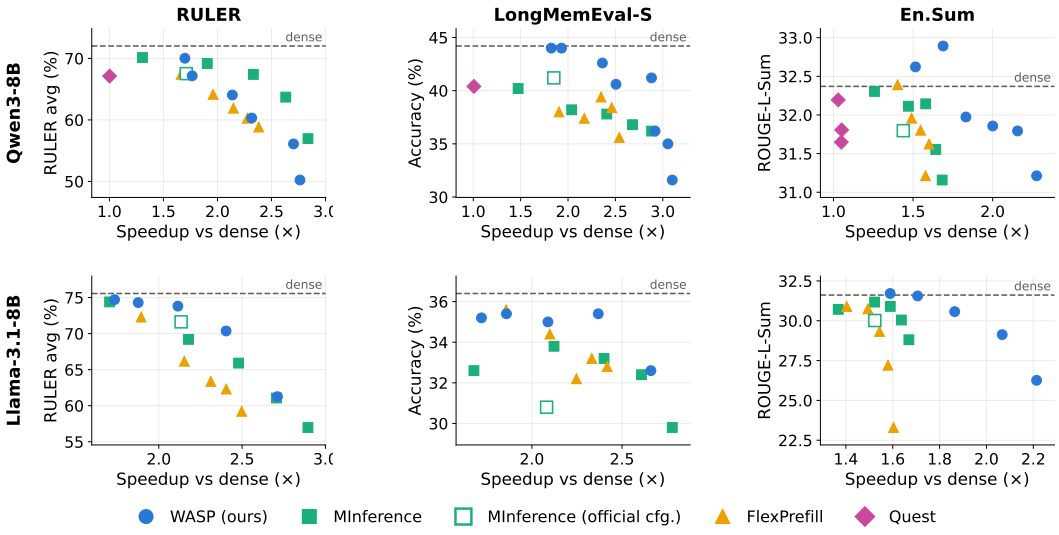


Figure 3: **Under question-agnostic prefilling, WASP leads on five of the six Pareto fronts, on the prefill-dominated LongMemEval for both models and RULER for Llama-3.1-8B, and on the decode-dominated ∞Bench En.Sum book summarization for both models.** We provide detailed benchmark results in Section G and the Pareto fronts under the standard protocol in Fig. 9 for reference.

MInference 2.6 points behind, and at τ=2.0 it runs faster than both baselines while staying within 0.6 points of the best baseline entry on both benchmarks.

The decode-heavy En.Sum isolates what the asymmetric design is worth: the prefill-only methods are bounded by the decoding share and stop at 1.3–1.7× and the decode-only Quest at 1.05×, while WASP continues to 2.2× on Qwen3-8B and 1.9× on Llama-3.1-8B within about a ROUGE point of dense. At the attention layer itself the gains are 3.9× in prefilling and 2.7× in decoding at τ=2.5, and they grow with context length (Fig. 4).

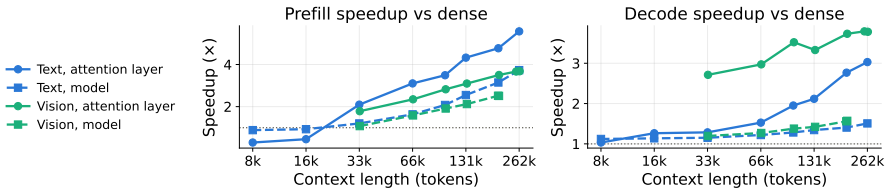


Figure 4: **WASP's speedup grows with context length in both phases and both modalities.** At τ=2.5 on text (Qwen3-8B, book prefixes) and vision (Qwen3-VL-8B, video prefixes, vision encoder excluded). Text prefilling breaks even between 16k and 33k tokens, and at the longest contexts the attention layer runs 3.0 to 5.6× faster than dense.

Table 2: **On Qwen3-VL-8B's video-understanding and OCR tasks, WASP matches dense accuracy at 1.1–2.0× speedup and changes by at most 0.4 points between protocols, whereas the question-guided SparseVLM and PyramidKV lose accuracy under the question-agnostic protocol.** Arrows represent the score change from *std* to *q-agn.*, and bold marks the best non-dense entry. †highest-accuracy OCR configuration (SparseVLM b=192, PyramidKV c=2048). Detailed results are in Section G.

Method	**Video-MME**			**LVBench**			**OmniDoc**	
	std	q-agn.	spd	std	q-agn.	spd	ED↓	spd
Dense (FA2)	72.3	72.3	1.00×	58.1	58.1	1.00×	0.158	1.00×
FastV R=0.5	71.9	71.9	1.38×	57.9	**58.1** ↑0.2	1.38×	0.177	1.01×
SparseVLM b=384	72.2	71.4 ↓0.8	**2.11**×	57.7	55.4 ↓2.3	**2.08**×	0.775†	1.01×
PyramidKV c=16384	**72.3**	69.0 ↓3.3	0.99×	**58.1**	49.4 ↓8.7	0.99×	0.461†	1.01×
WASP τ=2.5	72.1	**72.2** ↑0.1	1.86×	57.7	**58.1** ↑0.4	1.99×	**0.145**	**1.13**×

4.2 Vision tasks

WASP at τ=2.5 matches dense on all three tasks under both protocols, at 1.9–2.0× on video and 1.1× on OCR, where the 4.6k-token workload leaves little attention time for any method to save. Under question-agnostic prefilling SparseVLM gives up a further 0.8 and 2.3 points at its flagship budget and PyramidKV, whose compression needs the question present, up to 17 (Section G), while FastV moves by at most 0.2 points and WASP by at most 0.4. On the decode-heavy OCR the picture is different in kind, since generation must read the whole page and the pruning and eviction baselines lose most of their accuracy at every budget but FastV's mildest, while WASP stays at the dense edit distance.

5 Ablations

All ablations use Qwen3-8B on LongMemEval-S with 500 samples, and Section E gives the settings and the supporting measurements. The first question is whether IDW pooling is what makes block-query attention viable. Figure 6 swaps only the pooling function, and every alternative frontier is dominated by the IDW frontier. Mean and SVD pooling score 12.0 and 8.8 against 28.2 at the fast end and 15.2 and 13.8 against 37.4 at τ=1, with their best points below the IDW plateau. Min–max keys with IDW queries tie IDW at τ=0.5 and fall behind by 4 to 10 points at every wider budget. Table 3 swaps one side at a time. The query side is the more sensitive one, since replacing the query representative with the mean costs 7.8 points and with the block's leading singular directions 15.2, while min–max and rank-2 SVD on the key side cost 7.0 each. A query representative dominated by the block's common component picks the wrong blocks, and the exact attention that follows never sees the right keys, whereas a poor key representative misranks a single block. We keep the dot product as the similarity behind the density and leave other choices to future work.

The second question is whether selection needs a sampler at all. With IDW fixed, a rigid top-k at the same expected budget loses 15.2 points, because one k over-serves sharp heads and starves

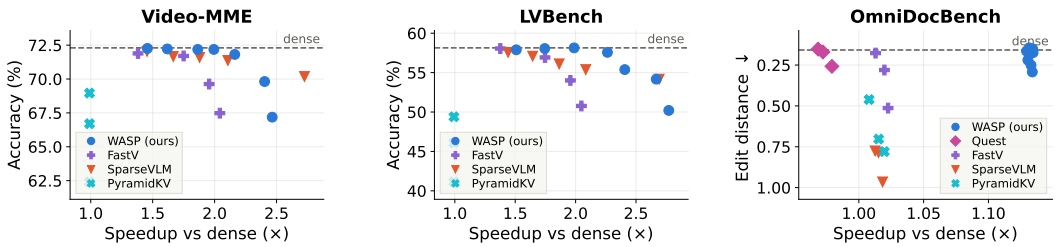


Figure 5: **WASP leads on all three multi-modal Pareto fronts, the prefill-dominated video understanding (Video-MME, LVBench) and the decode-dominated OCR (OmniDocBench).** We use Qwen3-VL-8B as the multi-modal base model, and the video understanding tasks use question-agnostic prefilling. Detailed results are in Section G and the standard-protocol fronts in Fig. 10.

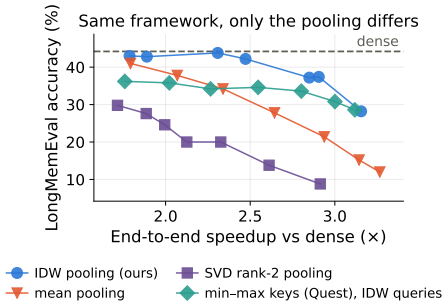


Figure 6: **WASP with IDW pooling leads the Pareto front against the same framework with mean, rank-2 SVD, or Quest's min–max pooling**, swept over temperatures on LongMemEval.

Table 3: One component swapped at a time on LongMemEval-S at τ=2.5, with IDW kept on the other side.

Variant	LME	Δ
WASP (IDW, Bernoulli)	**42.8**	—
query pooling (key side IDW)		
mean	35.0	−7.8
SVD rank-2	27.6	−15.2
key pooling (query side IDW)		
min–max (Quest)	35.8	−7.0
SVD rank-2	35.8	−7.0
selector (matched budget)		
top-k	27.6	−15.2
nucleus p=0.999	40.2	−2.6
sampler		
exact inclusion probability	41.2	−1.6
unbiased reweighting	42.8	0.0

diffuse ones, and nucleus selection recovers most of the gap but pays a sorting cost at every selection. Inside the sampler, the exact inclusion probability costs 1.6 points and unbiased reweighting changes nothing at this temperature, so we keep the linear form without the division (Section D). The uniform floor of Section 3.3 matters at the fast end, since, measured against the same kernels without it, the prefill floor adds 1.6, 3.0, and 1.6 points on LongMemEval at τ=1.0, 1.5, and 2.0 and is within noise from τ=2.5 up, and on OmniDocBench the two floors lower the edit distance from 0.53 to 0.22 at τ=0.5, at a cost of up to 3% of prefilling throughput and nothing at decoding. The multiplier i in the keep probability is not a knob, since replacing it by a constant of 50k, 200k, or 400k at τ=2.5 moves accuracy by at most 0.4 points.

6 Conclusion

We presented **W**eighted **A**symmetric **Sp**arse (WASP) attention, a training-free framework that sparsifies queries where compute dominates and keys where bandwidth dominates, with IDW pooling and a sorting-free Bernoulli sampler shared by both phases. On Qwen3-8B, Llama-3.1-8B, and Qwen3-VL-8B it stays within about two points of dense accuracy on five of six long-context benchmarks at about 2× measured end-to-end speedup, it is the only method in our comparison whose speedup continues on decode-heavy workloads, and it moves by at most 2 points under question-agnostic prefilling, where the probe-based baselines lose up to 7 points.

Our evaluation covers two text model families and one vision model at the 8B scale on a single H200. The speedup needs long inputs, with break-even between 16k and 33k tokens, and on RULER MInference keeps more retrieval accuracy at comparable speed on Qwen3-8B, a gap we attribute to pooled representatives that low-pass positional structure. WASP reduces compute and bandwidth but not KV-cache memory.

REPRODUCIBILITY STATEMENT

All accuracy numbers are full-benchmark runs (no subsampling beyond the official protocols), except LVBench, where unavailable source videos leave 1,235 of its 1,549 questions (Section J). All table cells are generated programmatically from the archived per-sample results, and each figure has a generation script. Code, kernels, evaluation harnesses, and raw measurement files will be released upon acceptance.

AI USE STATEMENT

We used an LLM-based coding assistant (Claude Code) throughout this work. Among the tasks that require disclosure, it was used to implement methods (the Triton kernels, the baseline integrations, and the evaluation harnesses), to help design and run the measurement protocol and the experiments under the authors' direction, and to produce first-pass summaries of results that the authors then interpreted. It was not used to generate synthetic data, to develop the conceptual framework, to formulate or prove mathematical claims, to propose the hypotheses, for translation, or for qualitative data analysis. Among the recommended disclosures, it generated the figure and table scripts, drafted and edited parts of the text for readability, and helped search the literature. All generated code was reviewed and tested by the authors, every number in the paper is produced programmatically from archived per-sample results that the authors inspected, and the authors take responsibility for the full content of the paper.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Federico Barbero, Alex Vitvitskyi, Christos Perivolaropoulos, Razvan Pascanu, and Petar Veličković. Round and round we go! what makes rotary positional encodings useful?, 2025. URL https://arxiv.org/abs/2410.06205.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=ayi7qezU87.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXXXI*, pp. 19–35, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-73003-0. doi: 10.1007/978-3-031-73004-7_2. URL https://doi.org/10.1007/978-3-031-73004-7_2.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. MagicPIG: LSH sampling for efficient LLM generation. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 44169–44190, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/6d50d824ae819d5a961c1d8edc15e833-Paper-Conference.pdf.

Yuan Cheng, Fengzhuo Zhang, Yunlong Hou, Cunxiao Du, Chao Du, Tianyu Pang, Aixin Sun, and Zhuoran Yang. Demystifying the slash pattern in attention: The role of roPE. In *ICML 2026 Workshop on Foundations of Deep Generative Models: Understanding Memorization, Generalization, and Reasoning*, 2026. URL https://openreview.net/forum?id=hJHQ2EQULj.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 35549–35562, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/98ed250b203d1ac6b24bbcf263e3d4a7-Paper-Conference.pdf.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359. Curran Associates, Inc., 2022. doi: 10.52202/068431-1189. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/67d57c32e20fd0a7a302cb81d36e40d5-Paper-Conference.pdf.

Tri Dao, Daniel Haziza, Francisco Massa, and Grigory Sizov. Flash-decoding for long-context inference. Stanford CRFM blog, October 2023. URL https://crfm.stanford.edu/2023/10/12/flashdecoding.html.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue

Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. DeepSeek-v3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

Rishi Desai, Jesse Hu, Joan Cabezas, Neel Harsola, Pratyush Shukla, Roey Ben Chaim, Adnan El Assadi, Omkaar Mukund Kamath, Fenil Faldu, Prannay Hebbar, Jiankai Sun, Yiyuan Li, Pramod Srinivasan, Ishan Gupta, Christopher Settles, Daniel Wang, Derek Chen, Pranav Raja, Albert Liu, Marek Šuppa, Nevasini Sasikumar, Luyang Kong, Erik Quintanilla, Xiangyi Li, Ivan Bercovich, and Steven Dillmann. SWE-marathon: Can agents autonomously complete ultra-long-horizon software work?, 2026. URL https://arxiv.org/abs/2606.07682.

Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective l_2 norm-based strategy for KV cache compression. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18476–18499, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1027. URL https://aclanthology.org/2024.emnlp-main.1027/.

Qihang Fan, Huaibo Huang, Zhiying Wu, Juqiu Wang, Bingning Wang, and Ran He. FlashPrefill: Instantaneous pattern discovery and thresholding for ultra-fast long-context prefilling, 2026. URL https://arxiv.org/abs/2603.06199.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, Ran He, and Xing Sun. Video-MME: The first-ever comprehensive evaluation benchmark of multi-modal LLMs in video analysis. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24108–24118, 2025. doi: 10.1109/CVPR52734.2025.02245.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, Clément Farabet, Johan Ferret, Filippo Galgani, Sertan Girgin, Jean-Bastien Grill, Maarten Grootendorst, Jiaxian Guo, Cassidy Hardin, Yanzhang He, Steven M. Hernandez, Omri Homburger, Léonard Hussenot, Juyeong Ji, Armand Joulin, Aishwarya Kamath, Parnian Kassraie, Olivier Lacombe, Preethi Lahoti, Gaël Liu, Gus Martins, Luciano Martins, Tatiana Matejovicova, Ramona Merhej, Nikola Momchev, Sneha Mondal, Ryan Mullins, Sindhu Raghuram Panyam, Shreya Pathak, Sarah Perrin, André Susano Pinto, Etienne Pot, Angéline Pouget, Alexandre Ramé, Sabela Ramos, Douglas Reid, David Rim, Morgane Rivière, Karsten Roth, Louis Rouillard, Omar Sanseviero, Pier Giuseppe Sessa, Shane Settle, Danila Sinopalnikov, Sara Smoot, Piotr Stanczyk, Andreas Steiner, Lawrence Stewart, Ilya Tolstikhin, Michael Tschannen, Anton Tsitsulin, Nino Vieillard, Renjie Wu, Pingmei Xu, Haichuan Yang, Edouard Yvinec, Biao Zhang, Li Zhang, Joe Zou, Nicolas Aagnes, Abdelrahman Abdelhamed, Jakub Adamek, Shivani Agrawal, Shubham Agrawal, Ibrahim Alabdulmohsin, Jean Baptiste Alayrac, Uri Alon, Chandramouli Amarnath, Ankesh Anand, Chrysovalantis Anastasiou, Setareh Ariafar, François-Xavier Aubet, Kyriakos Axiotis, Federico Barbero, Joelle Barral, Alexei Bendebury, Urs Bergmann, Stanley Bileschi, Kat Black, Mathieu Blondel, Sebastian Borgeaud, Arthur Bražinskas, Ryan Burnell, Robert Busa-Fekete, Mu Cai, Daniele Calandriello, Glenn Cameron, Charlotte Caucheteux, Rahma Chaabouni, Garima Chadha, Jetha Chan, Blake Jianhang Chen, Jesse Chen, Lin Chen, Xu Chen, Derek Cheng, Tzu hsiang Chien, Nikolai Chinaev, Yi Chou, Zhaohui Chu, Benjamin Coleman, Pooja Consul, Sam Conway-Rahman, Scott Crowell, Dylan Cutler, Vivek Dani, Samira Daruki, Anil Das, Daniel Deutsch, Nishanth Dikkala, Li Ding, Qiuhan Ding, Shenil Dodhia, Konstantin Donhauser, Tulsee Doshi, Anca Dragan, Alex Druinsky, Sahil Dua, Zoltan Egyed, Danielle Eisenbud, Daniel Eppens, Cindy Fan, Bahare Fatemi, Yassir Fathullah, Vlad Feinberg, Milen Ferev, Sebastian Flennerhag, Takumi Fujimoto, João Gabriel Oliveira, Isaac Galatzer-Levy, João Gante, Simon Geisler, Soham Ghosal, Antonious M. Girgis, Tamara von Glehn, Alec Go, Alhaad Gokhale,

Alex Grills, Yiming Gu, Mayank Gupta, Pramod Gupta, Guru Guruganesh, Raia Hadsell, Hamza Harkous, Jitendra Harlalka, Demis Hassabis, Anja Hauth, Joe Heyward, Arian Hosseini, Chih-Yang Hsia, I-Hung Hsu, Xiaopeng Huang, Yangsibo Huang, Kevin Hui, Adrian Hutter, Te I, Fotis Iliopoulos, Advait Jain, Ganesh Jawahar, Ziwei Ji, Qilin Jin, Melvin Johnson, Kandarp Joshi, Arun Kandoor, Wang-Cheng Kang, Koray Kavukcuoglu, Mehran Kazemi, Kathleen Kenealy, Amr Khalifa, Phoebe Kirk, Ivan Korotkov, Suraj Kothawade, Vitaly Kovalev, Neel Kovelamudi, Adam Kraft, Ravin Kumar, Vivek Kumar, Harish Kuppam, Justin Lannin, Chen-Yu Lee, Seungji Lee, Dmitry Lepikhin, Alon Levkovitch, Dongdong Li, Qiujia Li, Valentin Liévin, Ethan Lin, Ziqian Lin, Casper Liu, Tianlin Liu, Tianqi Liu, Xin Liu, Ivan Lobov, Mayank Lunayach, Min Ma, Gagan Madan, Andrii Maksai, Eric Malmi, Michal Matuszak, Daniel McDuff, Gaurav Menghani, Maciej Mikuła, Daniil Mirylenka, Karolis Misiunas, Vedant Misra, Andreea Mitran, Kareem Mohamed, Maksim Mukha, Eric Noland, James O'Donnell, Brendan O'Donoghue, Kate Olszewska, Bernett Orlando, Wanqiong Pan, Rina Panigrahy, Unnati Parekh, Nicolas Perez-Nieves, Chunjong Park, Eric Paskie, Liqian Peng, Bryce Petrini, Slav Petrov, Jonas Pfeiffer, Bilal Piot, Martyna Plomecka, Siim Poder, Octavio Ponce, Arijit Pramanik, David Racz, Anish Rajan, Michelle Ramanovich, Anand Rao, Marvin Ritter, Vitor Rodrigues, Evan Rosen, Mikołaj Rybiński, Noveen Sachdeva, Michaël E. Sander, Rohit Sathyanarayana, Sagar Savla, Samuel Schmidgall, Tal Schuster, George Scrivener, Benoit Seguin, Andrew Sellergren, Aliaksei Severyn, Izhak Shafran, Dhruv Shah, Bobak Shahriari, Yuan Shangguan, Ashish Shenoy, Pradeep Shenoy, Rakesh Shivanna, Pauline Sho, Lucas Spangher, Wojciech Stokowiec, Tim Strother, Yao Su, Yinghao Sun, Mukund Sundararajan, Andrea Tacchetti, Mor Hazan Taege, Pouya Tafti, Jean Tarbouriech, Chetan Tekur, Shantanu Thakoor, Rahul Thapa, Madeleine Traverse, Lenart Treven, Tao Tu, Chien Te Tung, Çağlar Ünlü, Petar Veličković, Malini Pooni Venkat, Sagar Gubbi Venkatesh, Vidya Venkiteswaran, Francesco Visin, Alex Vitvitskyi, Kiran Vodrahalli, Weiyi Wang, Xin Wang, Tris Warkentin, Jan Wassenberg, John Wieting, Cindy Wu, Lechao Xiao, Hao Xu, Yuhui Xu, Fuzhao Xue, Arun Yadav, Jun Yan, Antoine Yang, Lin Yang, Ming-Hsuan Yang, Ziyu Ying, Jae Hyeon Yoo, Morteza Zadimoghaddam, Sajjad Zafar, Fred Zhang, Jiageng Zhang, Jianyi Zhang, Xiaofan Zhang, Chao Zhao, David Zhou, and Chen Zou. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana

Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv

Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 52481–52515. Curran Associates, Inc., 2024. doi: 10.52202/079017-1663. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5dfbe6f5671e82c76841ba687a8a9ecb-Paper-Conference.pdf.

Dalton Jones, Junyoung Park, Matthew Morse, Mingu Lee, Matthew Harper Langston, and Christopher Lott. QuoKA: Query-oriented KV selection for efficient LLM prefill. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 72559–72578, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/760fe89b5a5e5792b33f23a2db1b9fbf-Paper-Conference.pdf.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. FlexPrefill: A context-aware sparse attention mechanism for efficient long-sequence inference. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 963–989, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/03645743ea35690f30d795d6bac149a5-Paper-Conference.pdf.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970. Curran Associates, Inc., 2024. doi: 10.52202/079017-0722. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/28ab418242603e0f7323e54185d19bde-Paper-Conference.pdf.

Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-LLaVA: Learning united visual representation by alignment before projection. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 5971–5984, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.342. URL https://aclanthology.org/2024.emnlp-main.342/.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, and Lili Qiu. RetrievalAttention: Accelerating long-context LLM inference via vector retrieval. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 54358–54385. Curran Associates, Inc., 2025. doi: 10.52202/085713-1816. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/4e36d4049fb0fea195a8267c8dcd0824-Paper-Conference.pdf.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34892–34916. Curran Associates, Inc., 2023a. doi: 10.52202/075280-1516. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6dcf277ea32ce3288914faf369fe6de0-Paper-Conference.pdf.

Hong Liu, Yuan Cheng, Lin Niu, Yi Su, Yufei Xue, Anmin Liu, Guanghua Yu, and Jianchen Zhu. PIVOT: Efficient query-group indexing for token-level sparse attention, 2026. URL https://arxiv.org/abs/2607.24593.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 52342–52364. Curran Associates, Inc., 2023b. doi: 10.52202/075280-2279. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a452a7c6c463e4ae8fbdc614c6e983e6-Paper-Conference.pdf.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Yutao Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. MoBA: Mixture of block attention for long-context LLMs. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 17790–17815. Curran Associates, Inc., 2025. doi: 10.52202/085713-0602. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/19eae75beed66321d62272e794a9c2ac-Paper-Conference.pdf.

NVIDIA. *CUDA Programming Guide, Section 4.2: CUDA Graphs*. NVIDIA Corporation, version 13.4.2 edition, 2026. URL https://docs.nvidia.com/cuda/cuda-programming-guide/04-special-topics/cuda-graphs.html. Accessed 2026-09-25.

Jungsuk Oh, Hyeseo Jeon, Hyunjune Ji, Kyongmin Kong, and Jay-Yoon Lee. Shallow prefill, deep decoding: Efficient long-context inference via layer-asymmetric KV visibility, 2026. URL https://arxiv.org/abs/2605.06105.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. OmniDocBench: Benchmarking diverse PDF document parsing with comprehensive annotations. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24838–24848, 2025. doi: 10.1109/CVPR52734.2025.02313.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 118–132, 2024. doi: 10.1109/ISCA59077.2024.00019.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 01 1972. ISSN 0022-0418. doi: 10.1108/eb026526. URL https://doi.org/10.1108/eb026526.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=KzACYw0MTV.

Yves Tillé. *Sampling algorithms*. Springer, 2006.

Philippe Tillet, H. T. Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International*

Workshop on Machine Learning and Programming Languages, MAPL 2019, pp. 10–19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367196. doi: 10.1145/3315508.3329973. URL https://doi.org/10.1145/3315508.3329973.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020. URL https://arxiv.org/abs/2006.04768.

Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Ming Ding, Xiaotao Gu, Shiyu Huang, Bin Xu, Yuxiao Dong, and Jie Tang. LVBench: An extreme long video understanding benchmark. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22958–22967, 2025a. doi: 10.1109/ICCV51701.2025.02131.

Yi Wang, Xinhao Li, Ziang Yan, Yinan He, Jiashuo Yu, Xiangyu Zeng, Chenting Wang, Changlian Ma, Haian Huang, Jianfei Gao, Min Dou, Kai Chen, Wenhai Wang, Yu Qiao, Yali Wang, and Limin Wang. InternVideo2.5: Empowering video MLLMs with long and rich context modeling, 2025b. URL https://arxiv.org/abs/2501.12386.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

Haocheng Xia, Mihir Pamnani, Hanxi Fang, Supawit Chockchowwat, and Yongjoo Park. LazyAttention: Efficient retrieval-augmented generation with deferred positional encoding. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=M9kHwqreN9.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=bTHFrqhASY.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5e5fd18f863cbe6d8ae392a93fd271c9-Paper-Conference.pdf.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 14138–14148, 2021.

Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. XAttention: Block sparse attention with antidiagonal scoring. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=KG6aBfGi6e.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Qingyue Yang, Jie Wang, Xing Li, Yinqi Bai, Tong Xialiang, Huiling Zhen, Jianye Hao, Mingxuan Yuan, and Bin Li. Why attention patterns exist: A unifying temporal perspective analysis. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 58115–58143, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/5f3ebdc66dd46b217cbcc7e21c6ef156-Paper-Conference.pdf.

Jiayi Yuan, Cameron Shinn, Kai Xu, Jingze Cui, George Klimiashvili, Guangxuan Xiao, Perkz Zheng, Bo Li, Yuxin Zhou, Zhouhai Ye, Weijie You, Tian Zheng, Dominic Brown, Pengbo Wang, Markus Hoehnerbach, Richard Cai, Julien Demouth, John D. Owens, Xia Hu, Song Han, Timmy Liu, and Huizi Mao. BLASST: Dynamic BLocked attention sparsity via softmax thresholding. In A. Chowdhery and Z. Jia (eds.), *Proceedings of Machine Learning and Systems*, volume 8, pp. 843–859. MLSys, 2026. URL https://proceedings.mlsys.org/paper_files/paper/2026/file/c6ee784cbe46d854843e4c883a3321ef-Paper-Conference.pdf.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1126. URL https://aclanthology.org/2025.acl-long.1126/.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17283–17297. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. SpargeAttention: Accurate and training-free sparse attention accelerating any model inference. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=74c3Wwk8Tc.

Jintao Zhang, Kai Jiang, Chendong Xiang, Weiqi Feng, Yuezhou Hu, Haocheng Xi, Jianfei Chen, and Jun Zhu. SpargeAttention2: Trainable sparse attention via hybrid top-k+top-p masking and distillation fine-tuning, 2026. URL https://arxiv.org/abs/2602.13515.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞Bench: Extending long context evaluation beyond 100K tokens. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15262–15277, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.814. URL https://aclanthology.org/2024.acl-long.814/.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. SparseVLM: visual token sparsification for efficient vision-language model inference. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025b.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710. Curran Associates, Inc., 2023. doi: 10.52202/075280-1506. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6ceefa7b15572587b78ecfcebb2827f8-Paper-Conference.pdf.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 193–210, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/zhong-yinmin.

A Related Work

Training-free sparse attention, one phase at a time. Prefill-only selectors sparsify the query-tile loop: MInference (Jiang et al., 2024) with pre-fitted per-head vertical–slash patterns, FlexPrefill (Lai et al., 2025) with coverage-adaptive block selection, XAttention (Xu et al., 2025) with antidiagonal block scoring, FlashPrefill (Fan et al., 2026) with faster pattern discovery, and QuoKA (Jones et al., 2026) with a fixed top-k of individual keys scored by a subset of each prefill chunk's queries, a selector it reuses at decoding with the single query and without a block-level representative. Decode-only selectors keep the full cache and cut per-step reads: Quest (Tang et al., 2024) scores pages by min–max bounds, MagicPIG (Chen et al., 2025) samples via LSH with CPU offloading, and RetrievalAttention (Liu et al., 2025) uses ANN search. Eviction methods reduce memory rather than time, permanently discarding entries via sink-plus-window (Xiao et al., 2024b), accumulated statistics (Zhang et al., 2023; Liu et al., 2023b), observation-window scoring (Li et al., 2024), per-layer pyramidal budgets (Cai et al., 2025), or ℓ_2-norm filtering (Devoto et al., 2024). In VLMs, FastV (Chen et al., 2024) and SparseVLM (Zhang et al., 2025b) drop visual tokens in early layers, and our question-agnostic results (Section 4.1) show this family's reported accuracy depends strongly on the question being present at compression time. Among the methods we compare, none is designed around both bottlenecks, and Section 4.1 shows that this is decisive on decode-heavy workloads. Systems that combine a prefill-time and a decode-time sparsifier exist, and a prefill-only method can be stacked with a decode-only one, but such combinations run two mechanisms with two selection rules and inherit the failure modes of each. BLASST (Yuan et al., 2026) is a recent exception that serves both phases with one rule. It reuses the running row maximum of online softmax inside the FlashAttention loop and skips the exponentiation, the $\mathbf{PV}$ product and, at decoding, the value-block load for any KV tile whose block maximum falls more than a fixed margin below the running maximum. Every $\mathbf{QK}^\top$ tile is still computed and every key block is still read, so the skip removes only the softmax and value side of the attend, the paper reports $1.52\times$ prefilling and $1.48\times$ decoding at about 72% sparsity at the kernel level and $1.1\times$ end to end, and the threshold that yields a given sparsity scales with context length, so one threshold per model and phase is fitted on about 1000 RULER sequences. Its own experiments compose it with a prefill selector and a decode selector, and it could likewise run inside the selected-key attend of WASP, so we treat it as complementary rather than as a baseline. WASP serves both phases with one block-attention template and one pooling function.

Pattern-based vs. pattern-free. MInference, FlexPrefill and MMInference-style methods commit to a vocabulary of attention patterns built on vertical, slash, and grid structures and estimate their positions from probes; XAttention (Xu et al., 2025) drops the probe by scoring blocks with antidiagonal sums, but keeps the structural prior that antidiagonal regularity predicts block mass; the scoring GEMM is moreover quadratic and threshold-independent, capping its end-to-end speedup well below WASP's operating range (Section L). Slash structure itself is increasingly traced to RoPE (Barbero et al., 2025; Cheng et al., 2026; Yang et al., 2026), and our results show the estimation, not the patterns, is the fragile part (Section 4.1). WASP makes no pattern assumption; the same content-scored block selection serves text and video unchanged, which also spares it the per-model offline pattern calibration MInference requires (none exists for Qwen3; Section 4).

Block summaries and selection rules. Any block-granular method needs a per-block representative and a rule for how many blocks to keep. The representative is usually mean-style: per-dimension min–max bounds in Quest (Tang et al., 2024), mean-pooled block keys in MoBA (Lu et al., 2025), mean-pooled queries and keys gating SpargeAttn's block filter (Zhang et al., 2025a), learned block compression in NSA (Yuan et al., 2025), and cached representative tokens in InfLLM (Xiao et al., 2024a). On the query side, QuoKA (Jones et al., 2026) replaces the representative by the 16 queries of a 128-token chunk least similar to the chunk mean and takes the maximum of their cosine scores per key, so a chunk scores every key 16 times where BQA scores it once with an IDW representative of 1024 queries, and its budget is a fixed top-k of keys shared by all heads and layers. Concurrently with this work, PIVOT (Liu et al., 2026) mean-pools groups of four adjacent queries into a proxy that replaces the per-query scan of the DeepSeek Sparse Attention indexer, and reports that the mean proxy exceeds a first-query or last-query proxy by more than ten RULER points at 128k, which supports Section 2, yet its score is summed over indexer heads, its budget is a fixed top-k per query, and its accuracy falls as the group grows from four to sixteen queries under a fixed candidate budget, so it requires a model that already carries the indexer and does not transfer to the pre-trained models of Section 4. Section 5 shows that the mean and min–max variants lose exactly the distinctive vectors

IDW preserves. The budget rule is almost always rigid top-k (or a coverage threshold, Lai et al., 2025), which Section 2 shows cannot track the more than $\sim$100$\times$ per-head sparsity spread. Our linear Bernoulli sampler replaces it at a kernel cost comparable to top-k. BLASST (Yuan et al., 2026) avoids pooling altogether by thresholding the exact block maximum of $\mathbf{QK}^\top$, which is only possible because it computes that product for every tile, so its rule is a saving inside the attend rather than a selection before it, and its threshold is shared by all layers and heads while per-head sparsity emerges from the scores. Both components are drop-in replacements for the methods above. BKA shares with Quest (Tang et al., 2024) the query-aware selection of key blocks at decoding, and both keep the KV cache intact and read only the chosen blocks at each step. Quest represents a page by its per-dimension minimum and maximum keys, keeps a fixed top-k of pages at every step, and runs dense prefilling, whereas BKA represents a block by its IDW-pooled key, draws blocks with the Bernoulli sampler so that each head's budget follows its own score distribution, shares its two operators with BQA at prefilling, and reorders visual tokens at write time so that each block covers a spatially contiguous region (Section F). PagedAttention (Kwon et al., 2023) also divides the cache into fixed-size blocks, but its blocks are units of memory allocation that let non-contiguous physical memory hold one sequence and every block is still read at every step, whereas BKA's blocks are units of scoring that decide which keys are read at all, so BKA can be laid out on top of paged memory and our implementation pre-allocates the cache as such systems do.

Trainable architectures (Beltagy et al., 2020; Zaheer et al., 2020; Wang et al., 2020; Xiong et al., 2021; Yuan et al., 2025; Lu et al., 2025) build sparsity in at training time, and DeepSeek Sparse Attention (DeepSeek-AI et al., 2025) learns an indexer. SPEED (Oh et al., 2026) makes the two phases asymmetric in depth rather than in what is pooled. Prompt tokens are processed and cached through the first K of L layers only, while generated tokens traverse all layers and, above the cutoff, attend only to a BoS anchor and to earlier generated tokens, so the prompt is removed from the upper layers wholesale with no scoring or selection. The cutoff is one constant shared by all heads and inputs, the reported gains at 128k are 1.33$\times$ time to first token and 1.22$\times$ time per output token at the cutoff that preserves quality, and the policy relies on SPEED-aware fine-tuning or a LoRA epoch, since applying it post hoc to Llama-3.1-8B-Instruct at that cutoff drops HotpotQA exact match from 55.4 to 37.5. WASP leaves the model's KV visibility intact, needs no training, and sets a dynamic per-head budget inside every layer.

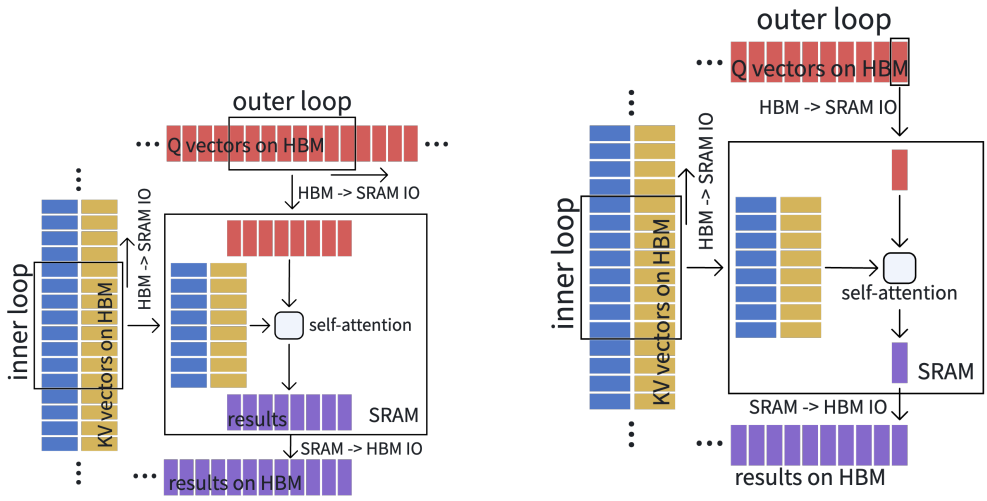
Systems. We build on FlashAttention (Dao et al., 2022; Dao, 2024; Dao et al., 2023), GQA (Ainslie et al., 2023), Triton (Tillet et al., 2019), and CUDA-graph decoding (NVIDIA, 2026). Serving systems (Kwon et al., 2023) and disaggregated prefill/decode (Zhong et al., 2024; Patel et al., 2024) motivate our per-phase analysis and the prefix-cache semantics of Section 4.1. Cross-request KV reuse is orthogonal to the within-request sparsity we target. LazyAttention (Xia et al., 2026) stores position-agnostic keys for retrieved documents and applies the RoPE offset inside the attention kernel, so one cached copy serves requests at any position, its gain comes from cache hits on chunks that recur across requests, its attend stays dense and exact, and its main evaluation uses a checkpoint fine-tuned for independent per-document encoding.

B FlashAttention bottleneck asymmetry

Figure 7 shows FlashAttention's tiled loops under the two phases, and Fig. 8 places both phases on the H200 roofline across context lengths.

C Coverage-budget measurement protocol

For a single head and query position with softmax weights $p_1, \ldots, p_S$ sorted descending, $k_\theta = \min\{k : \sum_{i=1}^{k} p_{(i)} \geq \theta\}$, $\theta \in \{0.9, 0.99, 0.999\}$. We run dense prefilling and record post-softmax weights of the last 64 query positions per layer (longest context, most representative of decoding). Per-layer and per-head figures use 10 LongMemEval samples at $\sim$100k on Qwen3-8B; the per-task figure uses six RULER subtasks; the per-modality figure uses Video-MME long videos on Qwen3-VL-8B with the visual-token mask separating query groups. Medians are taken across queries and samples.



(a) **Prefilling (compute-bound).** S/B_Q query tiles; total work $\mathcal{O}(S^2D)$ saturates tensor cores.

(b) **Decoding (bandwidth-bound).** A single query still streams the full KV cache each step.

Figure 7: FlashAttention's tiled loops under the two phases.

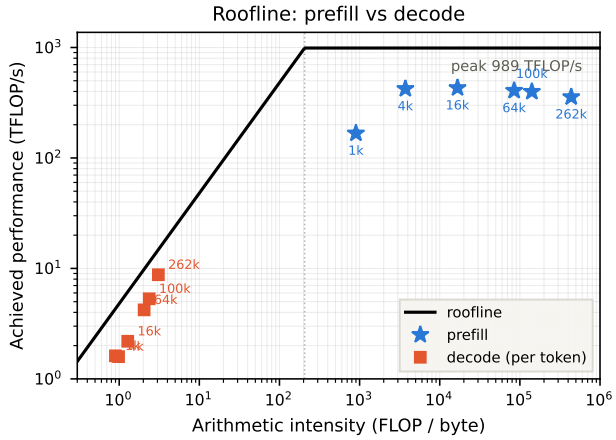


Figure 8: Roofline analysis (Qwen3-8B, H200) across context lengths. Prefilling (blue) is *compute*-bound and decoding (red) is *memory*-bound. The two phases sit on opposite sides of the ridge and call for different sparsification targets.

D LINEAR-APPROXIMATION BERNOULLI SAMPLER

Exact form. For a query with i preceding tokens and a distribution p over them, exact Bernoulli importance sampling keeps candidate j with $p_j^{\text{keep,exact}} = 1 - (1 - p_j)^i$ and reweights kept values by $1/p_j^{\text{keep,exact}}$ for unbiasedness.

Linear approximation. Expanding $1 - (1 - p_j)^i = ip_j - \binom{i}{2}p_j^2 + \mathcal{O}(p_j^3)$ and clamping gives $p_j^{\text{keep}} = \min(1, ip_j)$, accurate to $\mathcal{O}((ip_j)^2)$ when $ip_j \ll 1$, the dominant regime at $i \sim 10^5$.

Expected budget. $\mathbb{E}[|\mathcal{S}|] = \sum_j \min(1, ip_j)$ equals i for a flat distribution, where every preceding token is kept, and falls below it as the distribution sharpens, so the kept set follows per-head sparsity without a hard cutoff.

Stability. At 100k context p_j is of the order of 10^{-5} and smaller still on diffuse heads, where computing $\log(1-p_j)$ underflows and $1/(1-(1-p_j)^i)$ suffers catastrophic cancellation; the linear form has no such pathology. Empirically the biased linear sampler matches the unbiased one at the production temperature, 42.8 against 42.8 LME points (Table 3), because nearly every kept token has p^{keep} close to one there, and the exact inclusion probability costs 1.6 points.

Uniform floor. The floor of Section 3.3 raises every keep probability to at least U/i, so the kept set is the union of the score-driven draw and an expected U candidates drawn uniformly over the prefix, whatever the pooled score. Its role is coverage: the keys a block-level representative misses are spread over the whole prefix rather than near the block, so a uniform draw reaches them at a cost that grows with U and not with the context. Measured against the same kernels without it and with the same random keying, the prefill floor adds 1.6, 3.0, and 1.6 points on LongMemEval at τ=1.0, 1.5, and 2.0 and is within noise from τ=2.5 up, it is within noise on RULER and En.Sum at every temperature, and the decode floor changes nothing measurable on the text tasks. On the vision tasks the two floors add 2.2 and 2.4 points on Video-MME and LVBench at τ=0.5, and they lower the OmniDocBench edit distance from 0.53 to 0.22 at τ=0.5, from 0.32 to 0.18 at τ=1.0, and by 10 to 26% from τ=1.5 to 2.5, the one decode-heavy task in the suite. The floors cost up to 3% of text prefilling throughput, 0 to 2.5% of vision prefilling throughput, and nothing at decoding. Every WASP entry in the tables and figures runs with both floors on.

E Ablation details

Settings. All ablations use Qwen3-8B on LongMemEval-S with 500 samples and run the production configuration of the main results without the uniform floors, with the Triton block-query kernel and CUDA-graph decoding. The frontier comparison of Fig. 6 sweeps τ over $\{0.5, 1, 1.5, 2, 2.5, 3, 4\}$ for IDW, for the arithmetic mean and rank-2 SVD on both sides, and for min–max keys with IDW queries, since the query side has no min–max analogue, with speed from the same combined benchmark as the main results. The component swaps of Table 3 hold τ=2.5, the operating point of the main tables, and change exactly one component. The ablations predate the uniform floors of Section 3.3 and run without them, so the base row is a fresh run of that configuration rather than the floored entry of Table 5, and the budget perturbation below (42.4 to 43.0 around 42.8, and 43.8 for the repeat in Fig. 6) shows the run-to-run spread of the Bernoulli draw.

Pooling. The SVD variant replaces the representative by the block's leading singular directions (rank 2). It fails at least as badly as the mean because key blocks are strongly anisotropic, so a block's top singular direction nearly coincides with its mean direction, and different blocks' mean-style representatives share one dominant global component. Both summaries suppress the small block-specific residual that separates one block from another, which IDW preserves. The min–max and SVD representatives score with the temperatures of their own tuning, T=2 and T=10, since their upper-bound-style scores are peakier than the dot products of IDW and would otherwise keep far fewer tokens at the same τ.

Selection. Top-k sets k to the sampler's expected kept count at τ=2.5, measured on eight LongMemEval prompts as 13,328 keys per query block and head at prefilling and 1,385 key blocks per head at decoding. Nucleus selection keeps tokens in score order until a cumulative mass of p=0.999, which retains the long tail of diffuse heads and requires a sort at every selection, and it loses 2.6 points. The exact inclusion probability $1-(1-p_j)^i$ and the unbiased $1/p_j^{\text{keep}}$ reweighting are analyzed in Section D.

Budget. The multiplier i in the keep probability is the number of preceding tokens rather than a tuning knob, and the sampler is insensitive to its exact value. Replacing it by a constant of 50k, 200k, or 400k at τ=2.5 moves accuracy by at most 0.4 points (42.6, 43.0, 42.4 against 42.8 with the true count), so the temperature is the only knob the method exposes.

F Visual-KV spatio-temporal reordering

Qwen3-VL tokenises an (H, W)-patch image in raster order and a T-frame video as concatenated per-frame rasters. With block edge f and f^2=B_K (f=4) we permute the tokens of each frame at

cache-write time so that each consecutive B_K cache slots form one $f \times f$ tile of one frame:

$$\mathrm{slot}_{\mathrm{img}}(r, c) = \left\lfloor \frac{r}{f} \right\rfloor \frac{W}{f} f^2 + \left\lfloor \frac{c}{f} \right\rfloor f^2 + (r \bmod f) f + (c \bmod f),$$

and for video the same map is applied frame by frame with the frame index as the leading term, so a block never spans two frames. The permutation is fixed by the grid shape, applied once at write time, and free at decoding. Without it a "block" spans a horizontal strip of one frame and its representative carries no localized signal.

G FULL RESULT TABLES

Every point drawn in the Pareto figures appears here with its per-subtask detail. Tables 4, 5, 7 and 12 cover Qwen3-8B, Tables 8 to 11 cover Llama-3.1-8B, and Tables 2, 6 and 13 cover Qwen3-VL-8B.

Table 4: RULER (13 subtasks, 500 samples each, 100k context, Qwen3-8B), full sweeps. **Pref./Dec./Comb.** are measured prefilling, decoding, and end-to-end speedups vs. dense FlashAttention-2 with CUDA-graph decoding.

Method	cwe	fwe	ns_1	ns_2	ns_3	mk_1	mk_2	mk_3	mq	mv	qa_1	qa_2	vt	avg	Pref. tok/s	Dec. tok/s	Pref.	Dec.	Comb.
Dense (FA2)	51.1	90.0	100.0	100.0	99.8	77.4	39.0	7.2	91.5	96.4	57.2	46.2	80.4	72.0	9,082	72.7	1.00×	1.00×	1.00×
MInference official cfg.	56.3	88.7	100.0	100.0	99.8	79.8	31.6	7.0	90.8	95.1	57.8	42.8	78.8	71.4	15,831	72.7	1.74×	1.00×	1.71×
MInference s=6096	51.9	90.9	100.0	100.0	99.8	74.2	35.4	7.0	92.0	95.0	57.4	44.0	87.9	72.0	11,954	72.7	1.32×	1.00×	1.31×
MInference s=2048	61.8	89.5	100.0	99.8	99.8	75.2	34.2	6.0	91.6	94.0	57.6	41.4	84.7	72.0	17,716	72.7	1.95×	1.00×	1.91×
MInference s=1024	70.3	89.4	100.0	99.8	99.8	75.4	32.0	5.2	89.8	93.3	56.2	40.4	79.3	71.6	21,925	72.7	2.41×	1.00×	2.33×
MInference s=512	60.6	89.5	100.0	99.8	99.8	75.4	27.0	4.2	90.4	92.3	53.8	39.4	73.5	69.7	24,942	72.7	2.75×	1.00×	2.63×
MInference s=256	13.9	85.3	100.0	99.4	99.6	72.8	22.2	3.4	89.2	92.2	49.2	39.0	64.5	63.9	27,038	72.7	2.98×	1.00×	2.84×
FlexPrefill γ=0.95	34.0	90.5	100.0	99.8	98.8	79.4	29.2	7.8	93.0	95.3	59.4	44.8	85.9	70.6	15,379	72.7	1.69×	1.00×	1.66×
FlexPrefill γ=0.90	19.2	90.5	100.0	99.2	99.0	84.0	29.0	12.6	94.4	94.0	58.0	43.4	79.2	69.4	18,242	72.7	2.01×	1.00×	1.96×
FlexPrefill γ=0.85	11.2	90.8	100.0	99.4	97.2	86.6	29.8	13.8	94.3	91.7	52.4	42.2	74.6	68.0	20,094	72.7	2.21×	1.00×	2.15×
FlexPrefill γ=0.80	6.5	91.0	100.0	99.2	96.2	82.8	31.6	13.2	94.5	90.8	53.2	40.2	70.5	66.9	21,376	72.7	2.35×	1.00×	2.28×
FlexPrefill γ=0.75	4.6	91.1	100.0	99.4	96.0	78.8	29.8	16.4	94.1	88.5	50.2	41.8	67.5	66.0	22,426	72.7	2.47×	1.00×	2.38×
Quest b=4096	35.3	89.0	100.0	100.0	99.2	76.8	36.6	2.6	89.2	95.4	54.6	43.2	82.0	69.5	9,140	76.3	1.00×	1.04×	1.00×
Quest b=2048	23.8	88.8	100.0	99.6	98.6	76.4	32.6	1.6	86.2	90.7	51.8	41.4	81.1	67.1	9,147	80.0	1.00×	1.09×	1.00×
Quest b=1024	15.8	88.9	100.0	97.8	94.4	73.0	26.6	1.2	77.0	80.0	49.4	41.6	80.0	63.5	9,164	80.3	1.00×	1.09×	1.00×
WASP τ=4.0	37.0	89.9	100.0	100.0	99.8	72.6	41.4	6.6	90.7	93.6	57.2	44.6	73.4	69.8	15,594	87.1	1.72×	1.20×	1.70×
WASP τ=3.0	23.6	86.8	100.0	99.0	99.6	68.8	37.2	5.4	90.8	89.6	57.8	44.6	62.7	66.6	16,219	89.6	1.79×	1.23×	1.77×
WASP τ=2.5	17.8	85.7	99.8	98.0	99.4	66.2	31.0	4.0	90.0	86.5	56.6	43.4	48.6	63.6	19,734	95.2	2.17×	1.31×	2.14×
WASP τ=2.0	12.5	83.3	100.0	94.0	98.0	66.6	25.6	2.4	87.8	80.7	56.0	42.6	26.6	59.7	21,441	97.4	2.36×	1.34×	2.32×
WASP τ=1.5	8.3	82.4	99.8	92.2	95.8	64.0	19.2	1.2	81.2	69.9	56.4	38.6	14.0	55.6	25,134	103.2	2.77×	1.42×	2.70×
WASP τ=1.0	3.1	84.5	99.0	88.6	84.8	55.6	11.4	0.6	65.4	45.6	51.2	34.4	11.3	48.9	25,703	104.7	2.83×	1.44×	2.76×

H QUESTION-AGNOSTIC PREFILLING PROTOCOLS AND EQUIVALENCES

Text (context-only prefilling). The question span is located with per-task rules; step-1 prefills all tokens before it, step-2 prefills the question suffix (26–102 tokens across tasks; median 43) with dense attention over the cached context (serving systems process short suffixes densely), and generation proceeds unchanged. Gates: the dense two-step protocol is token-identical to one-shot on test samples (residual mismatches were traced to bf16 exact ties); WASP produced identical predictions on all smoke samples; Quest's full RULER score reproduced its one-shot value exactly (67.13), a constructive check of the whole pipeline.

En.Sum requires no rerun. The official template is "*Summarize the following book.*$\backslash n \backslash n\{$context$\}$". The instruction precedes the book, so the last 64/128 prefilling tokens (the estimation windows of MInference/FlexPrefill) contain only book text under the standard protocol already; a context-only split would move only $\sim$10 chat-control tokens.

Video (video-only prefilling). The shared prefix is capped at the final vision token (plus the vision-end boundary token, so SparseVLM's text-rater pool is non-empty); question and instruction prefill as a suffix per question. A middle rung that leaves the generic instruction in the shared prefix changes SparseVLM by at most 2 points; the question itself, not the instruction, carries the foreknowledge advantage.

Table 5: LongMemEval-S (500 samples, Qwen3-8B) by question type, full sweeps.

Method	know.	multi-	sess.	sess.	sess.	temp.	overall	Pref. tok/s	Dec. tok/s	Pref.	Dec.	Comb.
Dense (FA2)	69.2	22.6	78.6	20.0	80.0	23.3	44.2	8,568	71.6	1.00×	1.00×	1.00×
MInference official cfg.	64.1	18.8	67.9	20.0	77.1	21.1	40.2	16,378	71.6	1.91×	1.00×	1.85×
MInference s=6096	62.8	19.5	73.2	16.7	71.4	21.1	39.8	12,854	71.6	1.50×	1.00×	1.47×
MInference s=2048	62.8	20.3	73.2	13.3	70.0	18.0	38.8	18,180	71.6	2.12×	1.00×	2.04×
MInference s=1024	66.7	21.8	67.9	13.3	75.7	18.8	40.2	21,802	71.6	2.54×	1.00×	2.41×
MInference s=512	60.3	23.3	69.6	6.7	70.0	22.6	39.6	24,525	71.6	2.86×	1.00×	2.68×
MInference s=256	53.8	20.3	66.1	10.0	68.6	21.1	37.0	26,589	71.6	3.10×	1.00×	2.88×
FlexPrefill γ=0.95	61.5	26.3	83.9	16.7	75.7	24.1	44.0	16,897	71.6	1.97×	1.00×	1.90×
FlexPrefill γ=0.90	53.8	27.8	75.0	16.7	81.4	21.8	42.4	19,469	71.6	2.27×	1.00×	2.17×
FlexPrefill γ=0.85	53.8	22.6	78.6	13.3	75.7	21.8	40.4	21,205	71.6	2.48×	1.00×	2.35×
FlexPrefill γ=0.80	56.4	24.1	76.8	20.0	72.9	18.0	40.0	22,322	71.6	2.61×	1.00×	2.46×
FlexPrefill γ=0.75	55.1	24.8	69.6	16.7	61.4	26.3	39.6	23,120	71.6	2.70×	1.00×	2.54×
Quest b=4096	61.5	21.1	80.4	20.0	80.0	24.1	43.0	8,688	75.4	1.01×	1.04×	1.01×
Quest b=2048	61.5	20.3	73.2	13.3	72.9	21.8	40.0	8,660	79.0	1.00×	1.09×	1.01×
Quest b=1024	62.8	22.6	71.4	13.3	62.9	20.3	38.8	8,675	79.4	1.01×	1.10×	1.01×
WASP τ=4.0	62.8	21.1	75.0	30.0	82.9	26.3	44.2	15,908	88.6	1.86×	1.24×	1.82×
WASP τ=3.0	62.8	24.1	76.8	20.0	74.3	24.8	43.0	16,870	91.9	1.97×	1.28×	1.93×
WASP τ=2.5	61.5	21.8	73.2	23.3	75.7	24.1	42.0	20,839	96.4	2.43×	1.35×	2.36×
WASP τ=2.0	65.4	21.8	69.6	23.3	77.1	21.8	41.8	22,144	99.6	2.58×	1.39×	2.51×
WASP τ=1.5	57.7	22.6	67.9	16.7	74.3	21.1	39.6	25,667	102.5	3.00×	1.43×	2.88×
WASP τ=1.0	51.3	20.3	64.3	13.3	75.7	21.8	37.8	25,989	104.6	3.03×	1.46×	2.92×
WASP τ=0.75	41.0	18.8	62.5	10.0	68.6	19.5	33.8	27,281	105.2	3.18×	1.47×	3.05×
WASP τ=0.5	37.2	15.8	51.8	13.3	60.0	16.5	29.4	27,728	105.1	3.24×	1.47×	3.10×

I Implementation and engineering parity

Pre-allocated cache. Every KV buffer is allocated once before prefilling at its configured capacity, and nothing is allocated during inference. A cache that grows as prefilling proceeds pays a reallocation and a copy of everything already written each time it doubles, at a cost that grows with the context, and lazily allocated large tensors interleave with per-layer activations so the allocator cannot reuse pages cleanly. The decode-time buffers are views of the prefilling buffer, so finalising the cache reorders indices rather than duplicating the keys and values. Pre-allocation is also what makes graph capture legal, since a captured region may not allocate. Production serving stacks pre-allocate for the same reasons (Kwon et al., 2023).

Dense attends stay dense. Sparsity changes which tokens are read, not how attention is computed. The always-attended window goes through FlashAttention-2, the selected keys go through a block-tiled Triton kernel of our own that computes the same exact softmax attention, and the two outputs are merged through their log-sum-exp. No kernel in WASP belongs to a newer library generation than the FlashAttention-2 that the dense reference and every baseline run on. Swapping FlashInfer's FA3-class kernel into that attend moves model prefilling by between -1.5% and $+3.8\%$ across the τ sweep, so the reported speedups do not depend on it.

Fused kernels. The IDW density and pooling collapse into two Triton kernels that keep the Gram products in registers and fuse the row sum into the matmul epilogue, which runs at about 1.2 ms per layer, under 1% of model prefilling at 100k. The BKA decoding kernel scores, selects, gathers and attends in one launch, and it distributes a fixed 256 CTAs by work rather than by head, since heads select different numbers of blocks; the work-balanced form is $\sim 7\times$ faster than a per-head grid.

Graph capture. A decoding step touches 36 layers and several kernels per layer, so launch overhead is a visible share of the step at these shapes. Every decoding path in this paper, ours and every baseline's, runs as a single CUDA-graph replay, which requires the sampler's random draw to happen inside the captured region rather than through the host RNG.

The same engineering for the baselines. Each baseline runs in this harness, on the same pre-allocated cache, with the same FlashAttention backends and the same graph-captured decoding. Where a public implementation could not be captured we reimplemented the method and checked that its accuracy is unchanged, as for Quest (Section J). Every method, WASP included, is normalised to a dense reference that runs FlashAttention-2 with native grouped-query attention, and the dense prefilling references of the different harnesses agree within 1–3% on the text and video benchmarks.

Table 6: Multi-modal results (Qwen3-VL-8B), standard question-aware prefilling, full sweeps. PyramidKV MCQ accuracy equals dense by construction (Section J). Quest applies to OCR only.

Method	**Video-MME**		**LVBench**		**OmniDocBench**	
	acc↑	speedup	acc↑	speedup	ED↓	speedup
Dense (FA2)	72.3	—	58.1	—	0.158	—
FastV R=0.5	71.9	1.38×	57.9	1.38×	0.177	1.01×
FastV R=0.25	71.5	1.75×	57.1	1.75×	0.280	1.02×
FastV R=0.1	69.8	1.96×	53.8	1.96×	0.513	1.02×
FastV R=0.05	67.4	2.04×	51.0	2.05×	—	—
SparseVLM b=384	72.2	2.11×	57.7	2.08×	—	—
SparseVLM b=288	72.0	2.73×	57.8	2.69×	—	—
SparseVLM b=192	71.0	3.42×	55.2	3.44×	0.775	1.01×
SparseVLM b=128	71.0	3.56×	54.6	3.60×	0.786	1.02×
SparseVLM b=64	62.1	4.24×	43.3	4.27×	0.965	1.02×
PyramidKV c=16384	72.3	0.99×	58.1	0.99×	—	—
PyramidKV c=8192	72.3	0.99×	58.1	0.99×	—	—
PyramidKV c=4096	72.3	0.99×	58.1	0.99×	—	—
PyramidKV$_{\text{OCR}}$ c=2048	—	—	—	—	0.461	1.01×
PyramidKV$_{\text{OCR}}$ c=1024	—	—	—	—	0.703	1.02×
PyramidKV$_{\text{OCR}}$ c=512	—	—	—	—	0.780	1.02×
Quest b=2048	—	—	—	—	0.152	0.97×
Quest b=1024	—	—	—	—	0.170	0.97×
Quest b=512	—	—	—	—	0.258	0.98×
WASP τ=4.0	72.4	1.46×	57.8	1.51×	0.165	1.13×
WASP τ=3.0	72.0	1.62×	58.1	1.74×	0.158	1.13×
WASP τ=2.5	72.1	1.86×	57.7	1.99×	0.145	1.13×
WASP τ=2.0	72.1	1.99×	57.2	2.26×	0.150	1.13×
WASP τ=1.5	71.8	2.16×	55.6	2.41×	0.174	1.13×
WASP τ=1.0	69.8	2.40×	54.0	2.67×	0.178	1.13×
WASP τ=0.5	66.7	2.47×	50.3	2.77×	0.219	1.13×
WASP τ=0.25	—	—	—	—	0.251	1.13×
WASP τ=0.1	—	—	—	—	0.292	1.13×

What the comparison measures is therefore the selection mechanism rather than the engineering around it.

Where the remaining time goes. These choices put the comparison on top of an already optimised stack rather than an unoptimised reference. The dense attention we compare against runs at the memory roofline, reading the 409.6 MB of KV per layer at 100k in 126 μs, about 68% of the H200's peak bandwidth. On the sparse side the remaining cost is no longer attention, since prefilling attention falls to 68.3 ms per layer against a 25.9 ms projection floor that no attention method can remove, and a batch-one decoding step must stream 16.4 GB of weights whatever attention does. The end-to-end speedups we report are therefore bounded by the parts of the model WASP does not change.

J Measurement protocol and disclosures

Benchmark statistics. RULER: 13 subtasks, *500 samples per subtask*, 100k context. LongMemEval-S: 500 questions, 102–112k tokens, six question types. ∞Bench En.Sum: 103 book-summarization samples, inputs up to ∼750k tokens middle-truncated to 126k under the official protocol, ∼900 generated tokens. Video-MME: 2,700 questions, no-subtitle protocol, up to ∼220k visual tokens. LVBench: hour-long videos, 1,235 valid questions (subset disclosure below). OmniDocBench: 1,651 PDF pages, OCR-style generation with ∼4.6k-token prompts and ∼1.1k generated tokens.

Baseline configurations. MInference: uniform vertical–slash budgets (vertical 1000, slash s swept); Qwen3 has no official per-head pattern config, and vertical–slash both dominates the official assignments and subsumes A-shape, so we also run the released per-head pattern search on the released

Table 7: ∞Bench En.Sum full sweeps (103 samples, official protocol, 126k inputs, $\sim$900 generated tokens).

Method	ROUGE-L-Sum	Pref. tok/s	Dec. tok/s	Pref.	Dec.	Comb.
Dense (FA2)	32.4	7,472	67.5	1.00×	1.00×	1.00×
MInference official cfg.	31.8	16,137	67.5	2.16×	1.00×	1.44×
MInference s=6096	32.3	11,692	67.5	1.56×	1.00×	1.26×
MInference s=2048	32.1	17,126	67.5	2.29×	1.00×	1.47×
MInference s=1024	32.1	21,112	67.5	2.83×	1.00×	1.58×
MInference s=512	31.6	24,033	67.5	3.22×	1.00×	1.64×
MInference s=256	31.2	26,192	67.5	3.51×	1.00×	1.68×
FlexPrefill γ=0.95	32.4	15,123	67.5	2.02×	1.00×	1.40×
FlexPrefill γ=0.90	32.0	17,793	67.5	2.38×	1.00×	1.49×
FlexPrefill γ=0.85	31.8	19,822	67.5	2.65×	1.00×	1.55×
FlexPrefill γ=0.80	31.2	21,071	67.5	2.82×	1.00×	1.58×
FlexPrefill γ=0.75	31.6	22,035	67.5	2.95×	1.00×	1.60×
Quest b=4096	32.2	7,403	73.4	0.99×	1.09×	1.03×
Quest b=2048	31.6	7,400	76.9	0.99×	1.14×	1.05×
WASP τ=4.0	32.6	13,994	81.8	1.87×	1.21×	1.52×
WASP τ=3.0	32.9	16,392	87.6	2.19×	1.30×	1.69×
WASP τ=2.5	32.0	18,701	91.5	2.50×	1.36×	1.83×
WASP τ=2.0	31.9	21,683	96.0	2.90×	1.42×	2.00×
WASP τ=1.5	31.8	24,369	100.8	3.26×	1.49×	2.16×
WASP τ=1.0	31.2	26,593	104.3	3.56×	1.54×	2.28×

Table 8: RULER on Llama-3.1-8B (13 subtasks, 500 samples each, 100k context) under the standard protocol, with end-to-end speedup.

Method	cwe	fwe	ns_1	ns_2	ns_3	mk_1	mk_2	mk_3	mq	mv	qa_1	qa_2	vt	avg	Pref. tok/s	Dec. tok/s	e2e
Dense (FA2)	0.3	71.9	100.0	98.4	99.6	96.6	80.6	98.2	96.2	96.5	73.6	41.0	29.3	75.6	10,087	86.3	1.00×
MInference official cfg.	0.1	82.1	93.2	91.6	97.6	71.0	93.8	52.0	61.6	88.2	66.0	39.4	46.7	67.9	22,080	86.3	2.13×
MInference s=6096	0.2	73.7	100.0	99.4	99.6	93.2	78.6	97.0	91.1	95.2	70.6	41.6	51.0	76.2	17,468	86.3	1.70×
MInference s=2048	0.1	74.0	99.6	98.4	99.6	90.0	61.4	90.4	81.1	89.6	67.4	40.2	66.6	73.7	22,562	86.3	2.18×
MInference s=1024	0.0	71.2	95.6	93.6	99.6	80.4	40.2	67.4	78.4	88.2	66.0	36.4	64.8	67.8	25,852	86.3	2.48×
MInference s=512	0.0	69.1	96.0	66.6	98.6	62.8	39.4	42.2	74.2	85.0	62.4	31.4	64.6	60.9	28,372	86.3	2.71×
MInference s=256	0.0	68.9	87.2	26.4	97.2	48.2	40.6	11.4	56.5	73.0	49.2	25.0	45.5	48.4	30,487	86.3	2.90×
FlexPrefill γ=0.95	4.1	63.2	100.0	98.8	99.0	96.0	63.8	93.0	95.1	94.9	71.6	40.6	47.5	74.4	19,494	86.3	1.89×
FlexPrefill γ=0.90	0.6	59.9	100.0	98.2	99.2	89.6	68.4	82.2	87.8	87.5	60.4	41.2	37.6	70.2	22,288	86.3	2.15×
FlexPrefill γ=0.85	0.6	62.7	100.0	93.6	98.6	85.0	74.4	73.4	79.4	83.4	55.4	39.6	29.4	67.3	24,023	86.3	2.31×
FlexPrefill γ=0.80	0.8	65.7	99.8	86.2	93.2	69.4	62.6	63.4	68.5	68.8	54.2	34.8	36.6	61.8	25,037	86.3	2.40×
FlexPrefill γ=0.75	0.4	67.9	100.0	69.4	85.2	54.6	37.6	55.2	56.8	46.8	48.0	32.6	37.3	53.2	26,059	86.3	2.50×
WASP τ=3.0	0.1	73.3	100.0	98.8	99.6	96.6	74.4	90.4	95.9	94.7	71.4	41.0	31.0	74.4	17,641	108.9	1.73×
WASP τ=2.5	0.2	73.9	99.8	98.8	99.6	96.0	71.2	88.8	95.9	93.3	71.8	41.2	32.5	74.1	19,110	113.2	1.88×
WASP τ=2.0	0.1	75.3	99.8	99.0	99.6	95.4	68.8	81.8	95.6	91.5	71.4	39.0	36.1	73.3	21,597	117.3	2.11×
WASP τ=1.5	0.1	77.0	99.8	99.0	98.6	93.4	62.2	59.2	90.0	78.7	69.4	39.0	46.1	70.2	24,624	123.1	2.40×
WASP τ=1.0	0.1	75.9	98.6	96.2	92.2	86.0	43.8	26.6	68.2	48.1	62.8	35.8	60.1	61.1	27,907	124.1	2.71×

calibration data (one ∞Bench kv_retrieval sample at 30k tokens) and report the configuration it returns as a further point (official cfg.). FlexPrefill: γ swept at the released defaults, τ_{JS}=0.1 with a minimum budget of 1024, under which a head whose query-aware estimate agrees with its vertical–slash estimate falls back to the vertical–slash pattern, as in the released code. Quest: page 16, budget swept; its public implementation cannot be CUDA-graph captured (empirically verified), so we re-implemented the page-selection algorithm unchanged on our graph-captured kernel stack; accuracy is identical and decoding throughput reaches $\sim$75 tok/s at 100k under graph replay. FastV: K=3, keep-ratio swept. SparseVLM: pruning layers $\{2, 6, 15\}$, budget sweep below. PyramidKV: capacity swept. MagicPIG was also run, but its LSH sampling with CPU offloading was slower than dense at 100k context in our single-GPU setting and is omitted from the tables.

Speed protocol. Dedicated benchmarks per benchmark family: identical prompt sets across methods (matched samples), CUDA-graph decoding with pure-replay timing, warm-up excluded, vision-encode time separated. End-to-end speedup composes measured prefilling time and decoding rate with the benchmark's measured dense generation length. Text components are measured at 100–107k on RULER/LongMemEval prompts and, for En.Sum, on real 126k book prompts. The kernels that

Table 9: LongMemEval-S on Llama-3.1-8B (500 samples) by question type, standard protocol, with end-to-end speedup.

Method	know.	multi-	sess.	sess.	sess.	temp.	overall	Pref. tok/s	Dec. tok/s	e2e
Dense (FA2)	65.4	17.3	78.6	10.0	45.7	21.8	36.4	10,087	86.3	1.00×
MInference official cfg.	52.6	13.5	67.9	10.0	32.9	21.8	30.4	22,080	86.3	2.08×
MInference s=6096	60.3	17.3	80.4	13.3	51.4	24.8	37.6	17,468	86.3	1.68×
MInference s=2048	53.8	16.5	73.2	13.3	55.7	24.1	36.0	22,562	86.3	2.12×
MInference s=1024	57.7	18.8	78.6	6.7	62.9	24.1	38.4	25,852	86.3	2.40×
MInference s=512	55.1	18.0	71.4	3.3	51.4	27.1	36.0	28,372	86.3	2.61×
MInference s=256	51.3	14.3	75.0	10.0	51.4	25.6	34.8	30,487	86.3	2.78×
FlexPrefill γ=0.95	62.8	21.8	75.0	6.7	51.4	25.6	38.4	19,494	86.3	1.86×
FlexPrefill γ=0.90	59.0	18.8	60.7	6.7	42.9	22.6	33.4	22,288	86.3	2.10×
FlexPrefill γ=0.85	61.5	18.0	51.8	10.0	35.7	20.3	31.2	24,023	86.3	2.25×
FlexPrefill γ=0.80	50.0	17.3	48.2	3.3	28.6	15.0	26.0	25,037	86.3	2.33×
FlexPrefill γ=0.75	46.2	15.8	48.2	10.0	30.0	18.8	26.6	26,059	86.3	2.42×
WASP τ=3.0	62.8	15.8	75.0	10.0	47.1	20.3	35.0	17,641	108.9	1.72×
WASP τ=2.5	64.1	15.8	73.2	6.7	45.7	18.8	34.2	19,110	113.2	1.86×
WASP τ=2.0	59.0	21.8	73.2	13.3	47.1	20.3	36.0	21,597	117.3	2.09×
WASP τ=1.5	57.7	18.8	67.9	3.3	50.0	19.5	34.0	24,624	123.1	2.37×
WASP τ=1.0	56.4	14.3	60.7	10.0	45.7	20.3	31.8	27,907	124.1	2.66×

Table 10: ∞Bench En.Sum on Llama-3.1-8B (103 samples, official protocol), with end-to-end speedup.

Method	ROUGE-L-Sum	Pref. tok/s	Dec. tok/s	e2e
Dense (FA2)	31.6	8,325	80.6	1.00×
MInference official cfg.	30.0	21,913	80.6	1.52×
MInference s=6096	30.7	16,161	80.6	1.37×
MInference s=2048	31.2	21,874	80.6	1.52×
MInference s=1024	30.9	25,169	80.6	1.59×
MInference s=512	30.0	27,934	80.6	1.64×
MInference s=256	28.8	30,080	80.6	1.67×
FlexPrefill γ=0.95	30.9	17,307	80.6	1.40×
FlexPrefill γ=0.90	30.7	20,642	80.6	1.49×
FlexPrefill γ=0.85	29.3	22,838	80.6	1.54×
FlexPrefill γ=0.80	27.2	24,651	80.6	1.58×
FlexPrefill γ=0.75	23.3	25,944	80.6	1.60×
WASP τ=3.0	31.7	16,061	105.1	1.59×
WASP τ=2.5	31.6	17,801	109.9	1.71×
WASP τ=2.0	30.6	20,279	116.4	1.86×
WASP τ=1.5	29.1	24,266	122.4	2.07×
WASP τ=1.0	26.3	27,344	127.0	2.21×

carry the uniform floors were timed head to head, on the same prompts, against the kernels the archived WASP measurements were taken with (0.98 to 1.03× at prefilling, within 3% at decoding), and the WASP entries carry that measured ratio. The archived measurements were taken against a dense reference that expanded grouped keys before attention and ran 6% behind FlashAttention-2 with native grouped-query attention, and with FlashInfer's FA3-class kernel in WASP's selected-key attend. Both were re-timed in the same process on the same prompts, and every entry carries the measured ratios, so all methods are reported against the faster FlashAttention-2 dense and WASP on its own Triton kernel.

Attention-layer speedup. Table 14 isolates the attention stage itself at $L\approx$106k (real LongMemEval prompts, mean over all 36 layers). *Prefilling:* CUDA-event brackets around every attention module during a production prefilling, minus the projection floor (q/k/v/o GEMMs, norms, RoPE; 25.9 ms/layer, measured separately and identical across models), on vanilla FlashAttention-2 for dense and the production BQA path for WASP. *Decoding:* CUDA-graph replay timing of the attention core on the real per-layer cache state: `flash_attn_with_kvcache` over the full cache for

Table 11: Context-only prefilling on Llama-3.1-8B, all measured budgets. Dense is invariant by construction; WASP is re-run under both protocols and moves by at most 2 points.

Method	**RULER avg**			**LongMemEval**		
	one-shot	q-agn.	Δ	one-shot	q-agn.	Δ
MInference official cfg.	67.9	71.6	+3.7	30.4	30.8	+0.4
MInference s=6096	76.2	74.4	-1.9	37.6	32.6	-5.0
MInference s=2048	73.7	69.2	-4.5	36.0	33.8	-2.2
MInference s=1024	67.8	65.9	-1.9	38.4	33.2	-5.2
MInference s=512	60.9	61.1	+0.1	36.0	32.4	-3.6
MInference s=256	48.4	57.0	+8.6	34.8	29.8	-5.0
FlexPrefill γ=0.95	74.4	72.3	-2.1	38.4	35.6	-2.8
FlexPrefill γ=0.90	70.2	66.2	-4.0	33.4	34.4	+1.0
FlexPrefill γ=0.85	67.3	63.4	-4.0	31.2	32.2	+1.0
FlexPrefill γ=0.80	61.8	62.3	+0.5	26.0	33.2	+7.2
FlexPrefill γ=0.75	53.2	59.2	+6.0	26.6	32.8	+6.2
WASP τ=3.0	74.4	74.7	+0.3	35.0	35.2	+0.2
WASP τ=2.5	74.1	74.3	+0.2	34.2	35.4	+1.2
WASP τ=2.0	73.3	73.8	+0.5	36.0	35.0	-1.0
WASP τ=1.5	70.2	70.4	+0.2	34.0	35.4	+1.4
WASP τ=1.0	61.1	61.3	+0.1	31.8	32.6	+0.8

Table 12: Context-only prefilling on text, all measured budgets. Dense is invariant by construction (Section H); WASP is re-run under both protocols and moves by at most 2 points.

Method	**RULER avg**			**LongMemEval**		
	one-shot	q-agn.	Δ	one-shot	q-agn.	Δ
MInference official cfg.	71.4	67.5	-3.9	40.2	41.2	+1.0
MInference s=6096	72.0	70.1	-1.8	39.8	40.2	+0.4
MInference s=2048	72.0	69.2	-2.8	38.8	38.2	-0.6
MInference s=1024	71.6	67.4	-4.2	40.2	37.8	-2.4
MInference s=512	69.7	63.7	-6.0	39.6	36.8	-2.8
MInference s=256	63.9	57.0	-6.9	37.0	36.2	-0.8
FlexPrefill γ=0.95	70.6	67.4	-3.2	44.0	38.0	-6.0
FlexPrefill γ=0.90	69.4	64.2	-5.3	42.4	37.4	-5.0
FlexPrefill γ=0.85	68.0	61.9	-6.1	40.4	39.4	-1.0
FlexPrefill γ=0.80	66.9	60.3	-6.6	40.0	38.4	-1.6
FlexPrefill γ=0.75	66.0	58.8	-7.2	39.6	35.6	-4.0
WASP τ=4.0	69.8	70.0	+0.3	44.2	44.0	-0.2
WASP τ=3.0	66.6	67.2	+0.6	43.0	44.0	+1.0
WASP τ=2.5	63.6	64.1	+0.4	42.0	42.6	+0.6
WASP τ=2.0	59.7	60.3	+0.6	41.8	40.6	-1.2
WASP τ=1.5	55.6	56.1	+0.5	39.6	41.2	+1.6
WASP τ=1.0	48.9	50.2	+1.3	37.8	36.2	-1.6
WASP τ=0.75	—	—	—	33.8	35.0	+1.2
WASP τ=0.5	—	—	—	29.4	31.6	+2.2

dense (126 μs/layer, matching the HBM roofline) vs. the production fused BKA kernel (score, select, gather, and attend in one launch) for WASP.

LVBench subset. 20 of 103 source videos are permanently unavailable (platform takedowns), leaving 1,235 valid questions; all methods are evaluated on the identical subset.

SparseVLM budgets. Official retention schedules exist for budgets $\{192, 128, 64\}$; the lower-sparsity budgets $\{480, 448, 416, 384, 288\}$ use our extrapolated schedules (fractions of retained tokens per pruning stage, named by the same average-token convention), added so the method has operating points inside WASP's speed range.

Table 13: Video-only prefilling, all configurations (SparseVLM at its $\leq 3\times$ budgets).

Method	**Video-MME**		**LVBench**	
	acc↑	speedup	acc↑	speedup
WASP τ=4.0	72.3	1.46×	57.9	1.51×
WASP τ=3.0	72.2	1.62×	58.1	1.74×
WASP τ=2.5	72.2	1.86×	58.1	1.99×
WASP τ=2.0	72.2	1.99×	57.6	2.26×
WASP τ=1.5	71.8	2.16×	55.4	2.41×
WASP τ=1.0	69.8	2.40×	54.2	2.67×
WASP τ=0.5	67.2	2.47×	50.2	2.77×
SparseVLM b=480	72.0	1.46×	57.6	1.44×
SparseVLM b=448	71.7	1.67×	57.1	1.64×
SparseVLM b=416	71.6	1.88×	56.1	1.86×
SparseVLM b=384	71.4	2.11×	55.4	2.08×
SparseVLM b=288	70.2	2.73×	54.2	2.69×
FastV R=0.5	71.9	1.38×	58.1	1.38×
FastV R=0.25	71.7	1.75×	56.9	1.75×
FastV R=0.1	69.6	1.96×	54.0	1.96×
PyramidKV c=16384	69.0	0.99×	49.4	0.99×
PyramidKV c=8192	66.7	0.99×	46.1	0.99×
PyramidKV c=4096	62.4	0.99×	41.1	0.99×

Table 14: Attention-layer time and speedup at $L \approx 106$k (Qwen3-8B, H200, mean over 36 layers, real LongMemEval prompts). The BKA decoding number includes its scoring and selection steps, while the dense reference reads the full KV cache.

Config	**Prefilling attn / layer**		**Decoding attn / layer**	
	ms	speedup	μs	speedup
Dense (FA2)	266.5	1.00×	126.3	1.00×
WASP τ=0.5	26.8	9.95×	22.2	5.70×
WASP τ=1	28.9	9.21×	23.2	5.45×
WASP τ=1.5	37.1	7.18×	27.3	4.62×
WASP τ=2	49.7	5.36×	35.6	3.54×
WASP τ=2.5	68.3	3.90×	47.4	2.66×
WASP τ=3	86.3	3.09×	59.5	2.12×
WASP τ=4	112.6	2.37×	80.2	1.57×

PyramidKV on MCQ. With full prefilling attention and compress-on-write, the MCQ answer token is produced from unmodified prefilling logits; we verified $2{,}700/2{,}700$ identical predictions across capacities on Video-MME and therefore report dense accuracy for its video rows (its LVBench sweep would be a deterministic duplicate). Its accuracy-relevant regime is multi-token generation (OCR) and the question-agnostic semantics of Section 4.1.

K Standard-protocol Pareto fronts

Figs. 9 and 10 show the full sweeps under the standard protocol, where the question is part of the one-shot prefilling, for the text tasks on both model families and for the multi-modal tasks. The serving-semantics counterparts in the main text are the headline comparison, and the differences between the two are the question-foreknowledge effects discussed in Sections 4.1 and 4.2.

L XAttention's speedup ceiling

XAttention (Xu et al., 2025) scores 128-token blocks by the sum of their strided antidiagonals (one GEMM between feature-concatenated Q and K slices), followed by a softmax at the reshaped

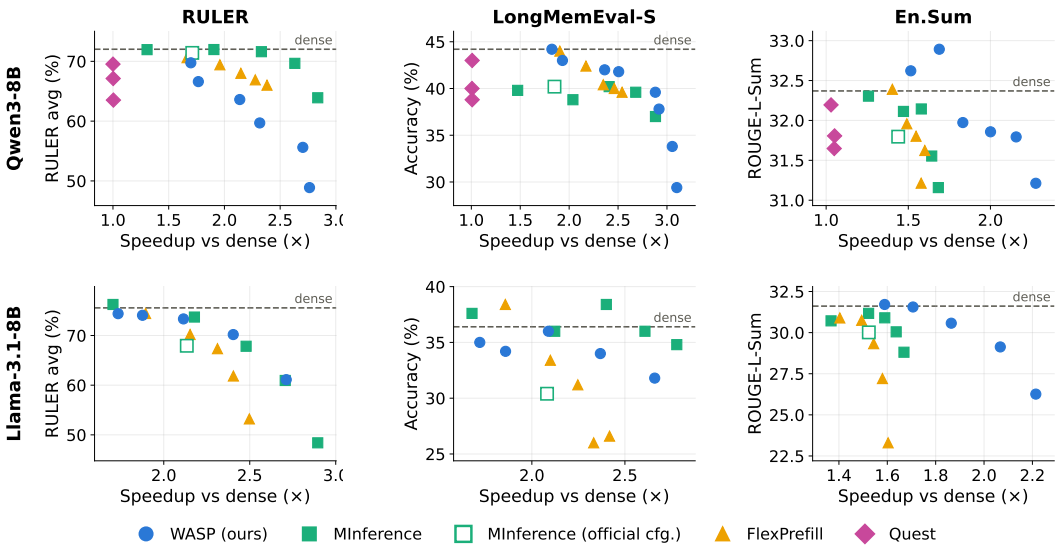


Figure 9: Text Pareto fronts under the standard protocol for Qwen3-8B (top) and Llama-3.1-8B (bottom). En.Sum is identical under both protocols.

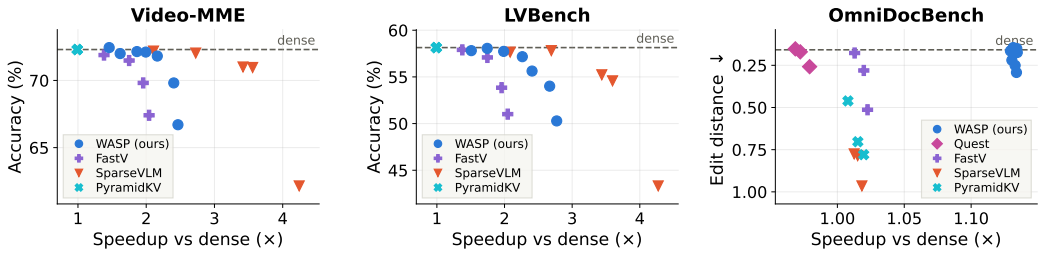


Figure 10: Multi-modal Pareto fronts for Qwen3-VL-8B under the standard protocol, where the question is visible during the video prefilling.

granularity, and keeps blocks by descending score until their cumulative mass reaches a threshold. We evaluate a validated reimplementation (block-mask agreement 0.996 with the official code on real Qwen3-8B activations; exact dense equivalence at threshold 1) with the official Block-Sparse-Attention kernel as the attend backend (output cosine 0.999998 to an exact gather-based attend). On LongMemEval it matches dense accuracy within noise at high thresholds, which no other prefill-only baseline we evaluate achieves. Its end-to-end speedup, however, cannot reach the range WASP operates in: sweeping the threshold from 0.95 all the way down to 0.5 (keeping only 3.9% of causal blocks) yields 1.11×–1.51× measured against the same-run dense reference at 100k.

The cap is structural, not a tuning artifact. The antidiagonal scoring costs $\Theta(S^2 D/\text{stride})$ with stride 8, one eighth of the full query–key product plus a reshaped-granularity softmax, and is *independent of the threshold*: it is paid in full before any block can be skipped. Measured at 100k layer shapes (Table 15), scoring alone costs 99 ms per layer against 211 ms for dense FlashAttention-2, and non-attention model time is 3.5 s (32% of the 11.1 s dense prefilling). Even with a zero-cost attend, end-to-end speedup could therefore not exceed 1.57×, and the measured threshold-0.5 point (1.51×) already sits at 96% of that ceiling. Substituting the official fused Triton scoring kernels raises the ceiling only to 1.79× (stride 8) / 1.84× (stride 16); the official kernels gate on the device *name* and route any non-"∗100" GPU to a slower fallback, so we bypass the string check on H200 (the same SM90 architecture as H100) to measure their fastest path. By contrast, WASP's pooled scoring costs $1/B_Q = 1/1024$ of the query–key product, two orders of magnitude less estimation overhead for the same cumulative-coverage selection family. This is why BQA reaches the 2.4× accuracy-matched operating point (τ=2.5) on the same benchmark and hardware. All numbers reproduce via `experiments/bench_xattn_ceiling.py`.

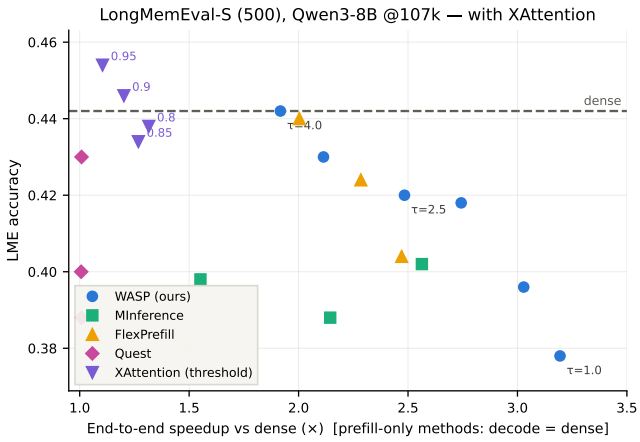


Figure 11: XAttention on LongMemEval (Qwen3-8B) across its threshold sweep, next to the other methods. Its accuracy is competitive, but its antidiagonal scoring is paid in full before any block is skipped, which caps its end-to-end speedup below the range WASP operates in.

Table 15: XAttention end-to-end speedup ceiling at 100k (Qwen3-8B, H200; dense FA2 attention 211 ms/layer, non-attention 3.5 s of the 11.1 s dense prefilling). The threshold-independent antidiagonal scoring plus non-attention time bound the speedup even if the sparse attend itself were free.

Scoring implementation	Scoring (ms/layer)	E2E ceiling (attend free)
Ours (PyTorch, stride 8)	99.1	1.57×
Official Triton, stride 8	75.2	1.79×
Official Triton, stride 16	70.3	1.84×