SPARSEDFLASH: DYNAMIC CONTEXT RETRIEVAL FOR LONG-VIDEO SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Block-diffusion drafters accelerate speculative decoding through parallel token prediction, but repeatedly attending to long video contexts can make drafting a bottleneck to overall speedup. Existing approaches reduce this overhead by restricting the draft context through sliding-window attention or visual token pruning. However, permanently discarding context can remove evidence needed for subsequent predictions. This limitation is particularly relevant to long videos, where successive predictions may depend on different, temporally distant events. We introduce SparseDFlash, an attention framework for efficient long-video speculative decoding that reuses the drafter's existing attention parameters without introducing additional learnable parameters. The framework preserves the complete context while dynamically restricting attention to evidence relevant to each drafting step. This allows the drafter to revisit distant evidence as generation progresses without repeatedly attending to the entire context. We further address a limitation of naive raster-order block construction, which can separate spatiotemporally adjacent visual tokens across retrieval blocks. Locality-aware ordering based on a three-dimensional Hilbert curve organizes these blocks according to the video's spatial and temporal structure. The framework preserves the target model's output distribution through standard speculative verification. Experiments on long-video description tasks using videos from four video understanding benchmarks show that SparseDFlash achieves state-of-the-art performance across all four benchmarks with both Qwen3-VL-8B and Qwen3.5-9B, while consistently outperforming the full-attention drafter using the same checkpoint.

1 INTRODUCTION

Speculative decoding (SD) (Leviathan et al., 2023; Chen et al., 2023) accelerates autoregressive generation by verifying multiple draft candidates in a single target-model pass. Its benefit depends on the target computation saved relative to the overhead of producing and verifying those candidates. Block-diffusion drafters such as DFlash (Chen et al., 2026) reduce drafting latency by predicting a block of tokens in parallel, conditioned on context features from the target model.

Long-video inputs introduce a substantial cost within each drafting step. Although the number of draft queries is fixed, full context attention requires them to attend to a growing number of visual keys and values. These cached states must be read repeatedly throughout generation, increasing both memory traffic and attention computation. Consequently, even a lightweight parallel drafter can incur substantial latency on long video contexts, limiting the end-to-end benefit of speculative decoding.

Sliding-window attention bounds this repeated context access by restricting attention to recent tokens. Visual speculative decoding methods also reduce drafting cost through token pruning or compression (Ji et al., 2025; Kang et al., 2025). For long-video understanding, however, the evidence needed during generation is not necessarily recent. Successive parts of a response may refer to different events, objects, or actions distributed throughout the video. A context budget allocated only by recency or fixed before generation may therefore exclude details needed by later predictions. Such omissions can reduce draft acceptance and offset the savings from cheaper attention.

Speculative decoding permits restricted context access during drafting while retaining the full context for target verification. DFlash additionally provides the drafter with cached context features

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.

extracted from the target model. These features allow the drafter to revisit earlier evidence if they remain available throughout generation. We introduce SparseDFlash, a sparse context attention framework that retains the complete draft context and dynamically selects blocks to read at each drafting step. The initial context and recent window remain accessible, while retrieved blocks supply evidence from more distant positions. Selection uses the drafter's existing attention projections, and blocks omitted at one step remain available for later retrieval. The target model continues to use its full context and standard verification procedure, preserving its output distribution.

Effective retrieval for parallel video drafting requires accounting for both the draft queries and the structure of the visual context. Sharing a retrieved block set across draft positions supports efficient KV access, but the selection must serve their different context needs. We use prefix-aware aggregation to combine routing scores across these positions, emphasizing early predictions because verification accepts only a prefix of the proposed tokens. Later queries still contribute evidence needed by subsequent predictions. For the visual context, frame-major serialization can place spatially or temporally neighboring tokens in different retrieval blocks. We therefore use locality-aware reordering based on a three-dimensional Hilbert traversal to construct blocks that better preserve these neighborhoods. Both refinements improve block retrieval without introducing learnable parameters or requiring additional training.

We evaluate SparseDFlash with Qwen3-VL-8B-Instruct and Qwen3.5-9B on video description workloads constructed from four video understanding benchmarks. Compared with full-attention DFlash using identical drafter weights, SparseDFlash improves average end-to-end speedup, with larger gains on benchmarks containing longer video contexts. We also apply sparse context attention to an official pre-trained DFlash checkpoint on long-context text tasks. The resulting improvements in acceptance and speedup demonstrate its applicability beyond video drafting without additional training. Our contributions are summarized as follows:

- We propose SparseDFlash, a sparse context attention framework for long-video speculative decoding that dynamically retrieves context while retaining the complete history for subsequent access.
- We introduce prefix-aware aggregation and locality-aware reordering to improve retrieval for parallel video drafting, accounting for prefix acceptance and spatiotemporal structure without additional parameters or training.
- We demonstrate that sparse context attention can be applied to block-diffusion drafters trained with full attention, allowing the same model to use dense or sparse context attention according to context length without additional training.

2 Related work

2.1 Speculative decoding

Speculative decoding accelerates language-model inference by verifying tokens proposed by a lightweight draft (Leviathan et al., 2023; Chen et al., 2023); verification is exact, so the target's output distribution is preserved at any temperature. Recent methods increase drafting parallelism through multiple prediction heads (Cai et al., 2024), feature-level drafts with tree-structured candidates (Li et al., 2024b; 2025), or diffusion-based generation (Arriola et al., 2025). Block-diffusion drafters such as DFlash (Chen et al., 2026) predict multiple response positions in parallel and have them verified in a single target-model pass, an arrangement Domino (Huang et al., 2026) extends with a causal encoder and a correction head. While these approaches reduce sequential drafting overhead, the draft model still attends to the full prefilling context, which becomes costly for long video inputs.

2.2 Video speculative decoding

Multimodal speculative decoding extends draft-and-verify inference to vision-language models. Existing approaches reduce drafting cost through modality-aware draft architectures (Ganesan et al., 2025), visual-token reduction before or during drafting (Kang et al., 2025; Ji et al., 2025), which builds on the broader token-reduction literature for VLM inference (Chen et al., 2024; Yang et al., 2025; Huang et al., 2025; Tao et al., 2025; Shao et al., 2025), selective or indirect use of visual

context (Zhang et al., 2026; Xie et al., 2026), or relaxed verification (Ji et al., 2026). These methods demonstrate that visual processing is an important source of drafting latency. They differ in what the drafter ends up seeing: SpecVLM and ViSpec fix a reduced visual input before generation, whereas Sparrow removes visual keys and values from draft attention and lets the drafter receive visual information indirectly through the target's text hidden states as generation proceeds. Our work instead addresses the drafter's long-context attention cost directly: the whole context stays stored and reachable, and each draft block retains a local window while dynamically retrieving a fixed number of remote blocks.

2.3 Sparse attention

Sparse-attention methods reduce long-context computation through local windows, blockwise compression, and query-dependent key-value selection (Xiao et al., 2024b; Zhang et al., 2023; Tang et al., 2024; Xiao et al., 2024a), with a more recent line making the sparsity native to training (Yuan et al., 2025; Lu et al., 2025; Gao et al., 2025; Zhao et al., 2026). Representative approaches use compact block representations, typically a mean over the block's keys, to estimate relevance and retrieve a small subset of full-resolution keys and values. Most existing methods redesign the attention mechanism of the primary language model for general long-context processing. In contrast, we apply sparse attention only into the speculative drafter, where the objective is to minimise draft latency while preserving target acceptance, and where the unit of selection is a jointly generated and prefix-verified draft block rather than an individual query. The proposed inference procedure keeps the pre-trained drafter frozen and also changes visual block membership through a geometric ordering of the video token grid.

3 Preliminaries

Speculative decoding accelerates an autoregressive target model by verifying multiple candidates from a lightweight draft model in parallel (Leviathan et al., 2023; Chen et al., 2023). Let τ denote the average number of tokens advanced per verification step, including the additional target token. The average decoding latency per generated token is

$$L_{\mathrm{SD}} = \frac{T_{\mathrm{draft}} + T_{\mathrm{verify}}}{\tau}, \tag{1}$$

where the two terms denote average drafting and verification time per step. This expression excludes prefill and separates the effects of draft latency and acceptance length.

DFlash (Chen et al., 2026) uses a block-diffusion drafter to predict masked future tokens in one forward pass. Each block contains one clean anchor token and P prediction positions, with bidirectional attention within the block. The drafter conditions on fused hidden features from selected target layers, which are projected into the keys and values of every draft layer. Full attention therefore reads the complete target context at every speculation step. We retain DFlash's parallel drafting and target verification, and replace this full context access with sparse retrieval while keeping the compatible pre-trained draft checkpoint frozen.

4 Method

SparseDFlash reduces the context accessed by a DFlash drafter through sparse context attention, prefix-aware block routing, and locality-aware reordering. It reuses the drafter's existing projections, introduces no learnable parameters, and requires no additional training. Target verification remains unchanged. The window size w, retrieval block size c, block budget K, and position-decay scale γ are fixed inference hyperparameters. Figure 1 provides an overview.

4.1 Sparse Context Attention

At an anchor position a, let $\mathcal{C}_{\mathrm{init}}$ denote the initial context block and $\mathcal{W}_a$ the recent context window, containing at most c and w tokens, respectively. We partition the stored context into retrieval blocks $\{\mathcal{C}_b\}$ of at most c tokens. The initial block and recent window provide persistent context, while additional blocks are selected dynamically according to the draft queries.

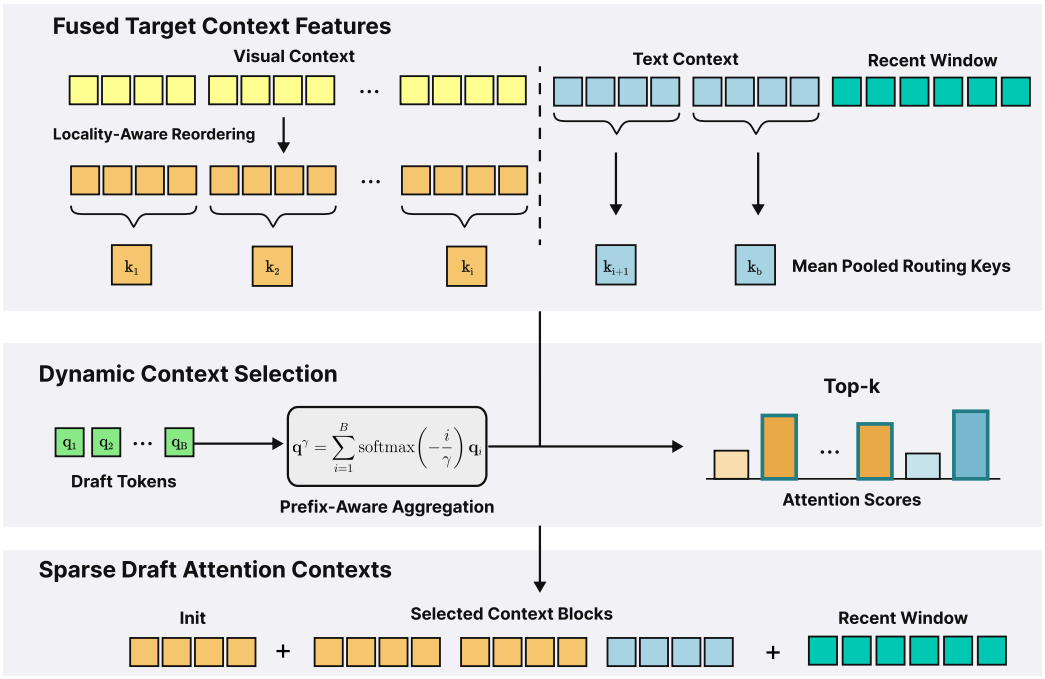


Figure 1: **Overview of SparseDFlash.** Locality-aware reordering groups visual tokens into blocks. Prefix-aware routing dynamically selects context blocks, which the drafter attends to together with the initial segment and recent window.

Let $\mathcal{R}_a$ denote the eligible remote blocks under the router's prefix and window constraints, excluding the initial block. Given a block budget K, we retain the initial block and select up to $K-1$ additional blocks $\mathcal{B}_a$:

$$\mathcal{M}_a = \mathrm{Concat}(\mathcal{W}_a, \mathcal{C}_{\mathrm{init}}, \{\mathcal{C}_b\}_{b\in\mathcal{B}_a}), \qquad |\mathcal{B}_a| = \min(K-1, |\mathcal{R}_a|). \tag{2}$$

Here, Concat denotes concatenation of the corresponding token lists. The budget K includes the initial block, so $K = 1$ disables remote retrieval. The selected context and current draft block share a single attention softmax, with bidirectional attention retained within the draft block.

Selection is performed independently in each draft layer. Query heads sharing a key-value head use a common block set, with a full-resolution context access budget of at most $w + Kc$ tokens per group, excluding the current draft block. Selected blocks are read at their original token resolution; pooling is used only to compute routing scores. All historical states remain stored, and routing scans the pooled block keys, so KV storage and routing cost still grow with context length.

Gathering the selected KV entries into a contiguous buffer before attention would introduce additional memory reads and writes at every draft step. We therefore fuse indexed KV access and attention computation in a Triton kernel. The kernel reads the window and selected blocks directly from the cache under a shared online-softmax state, avoiding an intermediate gathered KV buffer. Implementation details are provided in Appendix A.

4.2 Prefix-Aware Routing

Consider one draft layer at anchor position a and a group $\mathcal{G}$ of query heads sharing a key-value head. Let $\mathbf{k}_j$ denote the context key at token j and $\mathbf{q}_{i,h}$ the query at block position $i \in \{1, \ldots, P\}$ in head h. The P positions include the known anchor token at $i = 1$ and the remaining masked positions.

We represent each context block by its mean key and compute a relevance distribution over eligible remote blocks for each query position:

$$\bar{\mathbf{k}}_b = \frac{1}{|\mathcal{C}_b|} \sum_{j \in \mathcal{C}_b} \mathbf{k}_j,$$
$$p_{i,b} = \frac{1}{|\mathcal{G}|} \sum_{h \in \mathcal{G}} \mathrm{softmax}_{b \in \mathcal{R}_a} \left(\frac{\mathbf{q}_{i,h}^\top \bar{\mathbf{k}}_b}{\sqrt{d_{\mathrm{head}}}} \right). \tag{3}$$

Each head is normalized over the eligible blocks before its scores are averaged within the group.

Speculative verification accepts only a prefix of the proposed tokens: an error at an early candidate position prevents later candidates from being accepted. We therefore aggregate routing scores with exponentially decaying weights, emphasizing the anchor and early draft positions while retaining contributions from later queries:

$$r_b = \sum_{i=1}^{P} \omega_i p_{i,b}, \qquad \omega_i = \frac{\exp(-(i-1)/\gamma)}{\sum_{j=1}^{P} \exp(-(j-1)/\gamma)}, \tag{4}$$
$$\mathcal{B}_a = \mathrm{TopK}_{\min(K-1, |\mathcal{R}_a|)}(r).$$

The decay scale $\gamma > 0$ controls the emphasis on early positions. We set $\gamma = 7$, matching the decay scale used for position-wise loss weighting in DFlash training (Chen et al., 2026). The selected blocks are shared across query positions and heads within the group. When $K = 1$ or $\mathcal{R}_a$ is empty, remote scoring is skipped and $\mathcal{B}_a = \varnothing$.

4.3 Locality-aware Reordering

Text-oriented sparse attention methods, such as NSA Yuan et al. (2025) construct retrieval blocks from consecutive tokens. For video, however, sequence adjacency does not fully capture spatiotemporal locality. Frame-major raster order separates corresponding patches in adjacent frames by HW visual-token positions, while blocks crossing row or frame boundaries can contain spatially distant patches. Pooling these tokens into a single routing key can obscure local visual information. We therefore reorder visual tokens before block construction to group spatiotemporally neighboring patches.

Given a post-merge visual grid of size (T, H, W), we group C consecutive temporal slices and traverse each $C \times H \times W$ volume using a 3D Gilbert curve (Červený, n.d.), a generalization of the Hilbert curve Hilbert (1891) to rectangular grids. The traversal recursively subdivides the volume and adjusts the traversal direction within each subvolume to preserve locality across their boundaries. Consecutive tokens along this path thus tend to occupy nearby locations in both space and time (Figure 2). We concatenate the traversals in chronological group order and partition the resulting sequence into retrieval blocks of c tokens. Here, C controls the temporal span of each traversal, whereas c determines the retrieval block size. This construction encourages each mean-pooled routing key to summarize a local spatiotemporal region.

5 Experiments

5.1 Experimental Setup

Models and baselines. We evaluate SparseDFlash with Qwen3-VL-8B-Instruct and Qwen3.5-9B as main target vision language models and compare against DFlash (Chen et al., 2026), MSD (Lin et al., 2025), SpecVLM (Ji et al., 2025), and ViSpec (Kang et al., 2025). DFlash and SparseDFlash use the same underlying DFlash checkpoint; SparseDFlash enables sparse retrieval and Gilbert 3D reordering only at inference, with all weights frozen. SpecVLM is evaluated in the self-speculative decoding (Self-SD) setting and has no separately trained drafter. As the SpecVLM implementation does not support inference at temperature $T = 1$, we evaluate it only at $T = 0$.

Tasks and benchmarks. We evaluate speculative decoding on video captioning and video description tasks, which elicit long responses, using videos from VDC (Chai et al., 2025), MVBench

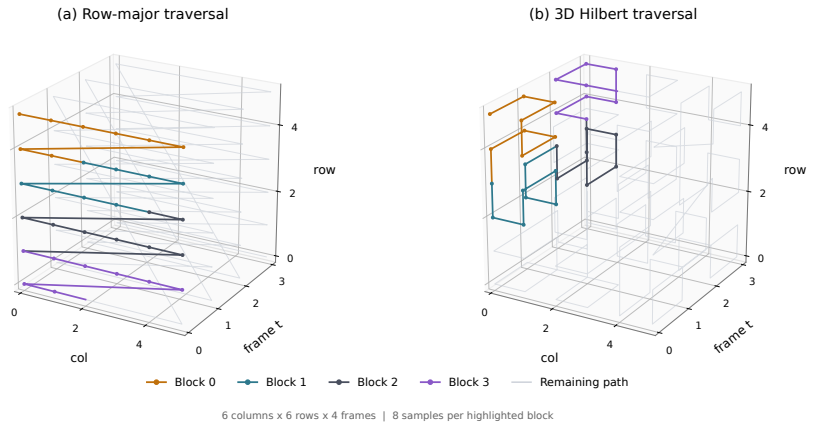


Figure 2: **Raster and Hilbert traversal on a** $6 \times 6 \times 4$ **grid.** Colors indicate consecutive groups of eight tokens, and gray denotes the remaining path.

(Li et al., 2024a), LongVideoBench (Wu et al., 2024), and Video-MME (Fu et al., 2025). The original multiple-choice prompts in MVBench, LongVideoBench, and Video-MME yield short answer selections, providing too few decoding steps to adequately assess decoding efficiency. Following the description-based evaluation protocol of SpecVLM (Ji et al., 2025), we use a common prompt across all four datasets: "Please provide a detailed description of the video, focusing on the main subjects, their actions, and the background scenes." For each dataset, we retain videos at least 60 seconds long and sample 50 videos uniformly across video durations, except MVBench which has total 32 vidoes over 60 seconds.

Training. For a fair comparison, we train the DFlash, MSD, and ViSpec drafters on the same data, with responses generated by the corresponding target model. We first train the DFlash drafter for three epochs on a mixture of video instruction data from LLaVA-Hound (Zhang et al., 2025a) and text conversations from ShareGPT. We then fine-tune it for 6,250 steps on LLaVA-Video (Zhang et al., 2025b), increasing the maximum training sequence length from 8,192 to 16,384 tokens. Both stages follow the DFlash training procedure, using full context attention, a draft block size of 16, and cross-entropy loss. We evaluate full-attention DFlash and SparseDFlash using the same checkpoint.

Implementation details. All inference experiments are conducted on a single NVIDIA B200 GPU. We set the video sampling rate to 2 fps, with a maximum of 768 frames per video. We evaluate at temperatures 0 and 1, with a maximum generation length of 512 tokens. We use a draft block size of 16 for both DFlash and SparseDFlash. We adjusts the maximum pixel budgets to 96Mi (about 49K tokens per video) to evaluate on long-video setting. Unless otherwise specified, SparseDFlash uses $w = 512$, $c = 32$, $K = 64$, and $\gamma = 7$.

Metrics. We report average acceptance length τ, including the additional target token, and end-to-end speedup relative to autoregressive decoding under the same target and sampling configuration.

5.2 Comparison with Baselines

Table 1 compares SparseDFlash with the baselines across four video benchmarks. SparseDFlash uses the same draft architecture and checkpoint weights as DFlash, changing only how the drafter accesses the context. On Qwen3-VL-8B, SparseDFlash achieves average end-to-end speedups of $2.269\times$ and $1.767\times$ at temperatures 0 and 1, respectively, compared with $1.924\times$ and $1.474\times$ for full-attention DFlash. On Qwen3.5-9B, the average speedup increases by 5.1% and 4.1%, respectively. These gains are obtained while maintaining average acceptance lengths close to those of DFlash on Qwen3-VL-8B and slightly improving acceptance on Qwen3.5-9B.

The advantage over full-attention DFlash is more pronounced on LongVideoBench and Video-MME, whose inputs are substantially longer than those of VDC and MVBench. On Qwen3-VL-8B,

Table 1: Average acceptance length (τ) and end-to-end speedup on Long-video benchmarks with Qwen3-VL-8B-Instruct and Qwen3.5-9B. Speedup is relative to autoregressive decoding on the same videos. DFlash and SparseDFlash shares same draft model weights.

Model	Method	VDC		MVBench		LongVideoBench		Video-MME		Avg.	
		τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup
Temperature = 0											
Qwen3-VL 8B	MSD	2.940	0.421×	2.748	0.413×	2.212	0.126×	2.350	0.358×	2.563	0.330×
	SpecVLM	**4.178**	0.740×	**4.039**	0.762×	**3.753**	0.367×	**3.846**	0.605×	**3.954**	0.619×
	ViSpec	2.897	0.574×	2.605	0.513×	1.966	0.236×	2.159	0.582×	2.407	0.476×
	DFlash	3.666	2.788×	3.231	2.439×	2.810	1.126×	2.921	1.342×	3.157	1.924×
	SparseDFlash	3.670	**2.843**×	3.221	**2.458**×	2.758	**1.823**×	2.870	**1.953**×	3.130	**2.269**×
Qwen3.5 9B	MSD	2.515	0.407×	2.483	0.399×	2.219	0.109×	2.394	0.280×	2.403	0.299×
	SpecVLM	**3.233**	0.321×	**3.101**	0.331×	**3.166**	0.239×	**3.240**	0.303×	**3.185**	0.299×
	ViSpec	2.615	0.732×	2.615	0.709×	2.318	0.235×	2.459	0.292×	2.502	0.492×
	DFlash	2.797	1.189×	2.940	1.240×	2.716	1.049×	2.673	1.064×	2.782	1.136×
	SparseDFlash	2.815	**1.201**×	2.947	**1.247**×	2.803	**1.146**×	2.716	**1.182**×	2.820	**1.194**×
Temperature = 1											
Qwen3-VL 8B	MSD	2.352	0.384×	2.079	0.321×	1.575	0.078×	1.766	0.330×	1.943	0.278×
	ViSpec	2.299	0.794×	1.821	0.634×	1.410	0.155×	1.572	0.545×	1.776	0.532×
	DFlash	**2.643**	2.093×	**2.434**	1.887×	**2.073**	0.878×	**2.183**	1.036×	**2.333**	1.474×
	SparseDFlash	2.590	**2.093**×	2.395	**1.921**×	2.042	**1.499**×	2.103	**1.555**×	2.283	**1.767**×
Qwen3.5 9B	MSD	2.257	0.380×	2.146	0.351×	1.897	0.100×	1.961	0.212×	2.065	0.261×
	ViSpec	2.319	0.679×	2.224	0.640×	1.918	0.207×	2.028	0.193×	2.122	0.430×
	DFlash	2.519	1.080×	2.484	1.064×	2.336	0.931×	**2.411**	0.956×	2.438	1.008×
	SparseDFlash	**2.527**	**1.088**×	**2.545**	**1.093**×	**2.344**	**1.011**×	2.364	**1.005**×	**2.445**	**1.049**×

SparseDFlash is $1.6\times$–$1.7\times$ faster than DFlash on LongVideoBench at both temperatures and $1.5\times$ faster on Video-MME. This pattern is consistent with the growing cost of full-context draft attention as input length increases. Qwen3.5-9B also shows larger gains on the longer benchmarks.

The lower speedups on Qwen3.5-9B partly reflect its higher verification overhead relative to single-token decoding. In our implementation, recurrent state updates in its GatedDeltaNet layers limit the efficiency of verifying a 16-token block compared with the full-attention target. This reduces the overall benefit of speculative decoding, even when draft attention is accelerated.

MSD, SpecVLM, and ViSpec accept multiple tokens per verification cycle, yet do not achieve end-to-end acceleration in our setting. This suggests that the savings from fewer target-model calls can be offset by the cost of producing and verifying the drafts. Their autoregressive drafting requires successive forward passes, each incurring computation and context access. These costs accumulate within each speculation cycle and can become substantial with long video inputs, explaining why acceptance gains need not translate into wall-clock speedups.

5.3 Ablation Studies

Effect of Dynamic Context Retrieval. We evaluate dynamic context retrieval with Qwen3-VL-8B on the Video-MME long split, varying the local window size $w \in \{256, 512, 1024\}$ and block budget $K \in \{1, 4, 16, 32, 64\}$. The same clips are evaluated under three context settings, denoted 32K, 64K, and 128K, with median prefill lengths of 35.9K, 71.7K, and 143.3K tokens, respectively.

Figure 3 shows that acceptance length increases monotonically with the remote block budget across all three context settings. Adding remote retrieval to a smaller local window yields higher acceptance than enlarging the local window alone. This suggests that access to distant evidence is more useful for draft prediction than restricting additional context access to recent tokens. At 64K, several

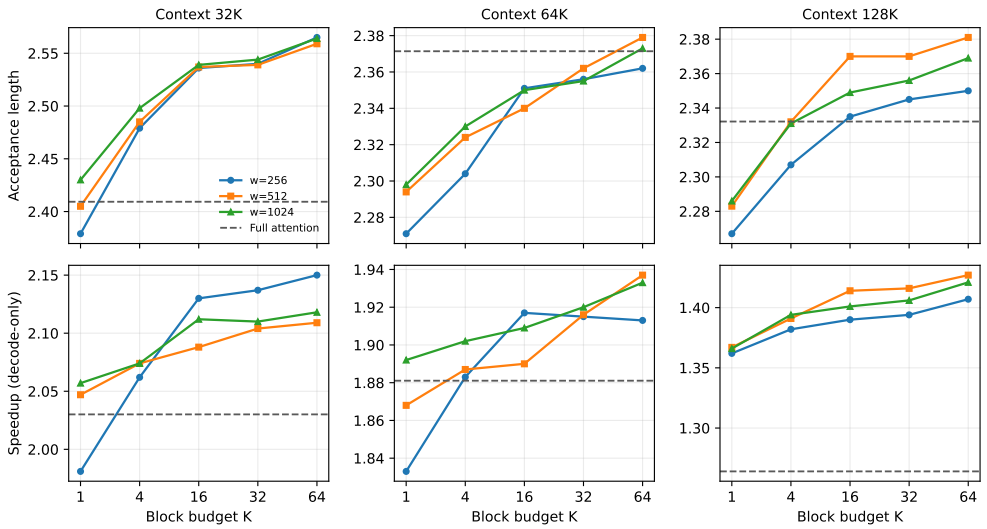


Figure 3: **Effect of dynamic context retrieval.** Acceptance length (top) and decode-only speedup over autoregressive decoding (bottom) for Qwen3-VL-8B on the Video-MME long split. Dashed lines denote full attention. The budget K includes the initial block; $K = 1$ disables remote retrieval.

Table 2: **Prefix aggregation.** Mean acceptance length on VDC. $\Delta\tau$ is the paired difference from prefix-aware aggregation, reported as mean $\pm$ standard error.

Aggregation	τ	$\Delta\tau$
Uniform	3.340	-0.242 ± 0.028
Anchor only	3.566	-0.016 ± 0.008
Prefix-aware	**3.581**	–

Table 3: **Visual block ordering.** Mean acceptance length on VDC. $\Delta\tau$ is the paired difference from raster order, reported as mean $\pm$ standard error.

Ordering	τ	$\Delta\tau$
Raster	3.552	–
Hilbert 3D ($C = 2$)	**3.581**	$+0.030 \pm 0.013$
Hilbert 3D ($C = 4$)	3.554	$+0.002 \pm 0.015$

sparse configurations achieve higher speedup than full attention despite lower acceptance, showing that reduced drafting cost can outweigh the reduction in accepted tokens per step.

With $K = 64$, the configurations using $w = 512$ or $w = 1024$ exceed full-attention acceptance across all three context settings. These results suggest that selectively accessing context can improve agreement between draft and target predictions, in addition to reducing attention cost. Detailed results are provided in Appendix B.

Prefix aggregation. On Qwen3-VL-8B, we compare anchor-only routing, uniform aggregation, and thre proposed prefix-aware aggregation with $\gamma = 7$ (Table 2). The weaker performance of uniform aggregation suggests that equally weighting all draft queries can dilute routing signals with less certain predictions from later positions. Restricting routing to the anchor query avoids this effect but may miss context needed by subsequent predictions. Prefix-aware aggregation incorporates queries from the entire draft block while assigning greater weight to early positions, whose correctness determines how much of the block can be accepted.

Visual block ordering. We compare frame-major raster order with 3D Hilbert ordering at temporal depths $C \in \{2, 4\}$ on Qwen3-VL-8B (Table 3). At $C = 2$, Hilbert ordering improves speedup without reducing acceptance length. Extending the traversal to $C = 4$ yields smaller gains, suggesting that grouping tokens across a wider temporal span does not necessarily improve block retrieval. Traversing pairs of adjacent frames achieves the best speedup among the evaluated settings.

Applicability to pre-trained drafters. We examine whether sparse context attention can be applied to existing block-diffusion drafters without additional training. We first evaluate the official

Table 4: **Applicability to a pre-trained text drafter.** Mean acceptance length (τ) and end-to-end speedup over autoregressive decoding on LongBench using the official Qwen3-8B DFlash checkpoint. Note that HotpotQA and Qasper do not have 32K context samples.

Context	Attention	HotpotQA		Qasper		GovReport	
		τ	Speedup	τ	Speedup	τ	Speedup
8K	Full	2.350	1.830×	2.097	1.663×	2.361	1.806×
	Sparse	**2.391**	**1.960×**	**2.240**	**1.897×**	**2.591**	**2.035×**
16K	Full	2.542	1.341×	1.566	0.894×	1.767	1.166×
	Sparse	**3.478**	**1.824×**	**1.949**	**1.086×**	**2.264**	**1.547×**
32K	Full	–	–	–	–	1.483	0.717×
	Sparse	–	–	–	–	**2.029**	**1.090×**

Table 5: **Selective application to a pre-trained drafter.** Mean acceptance length (τ) and end-to-end speedup over autoregressive decoding using the official Qwen3.5-9B DFlash checkpoint at temperature 0. Sparse attention replaces the final full-attention layer (last) or all draft attention layers (all), without additional training.

Benchmark	DFlash		SparseDFlash (last)		SparseDFlash (all)	
	τ	Speedup	τ	Speedup	τ	Speedup
VDC	3.392	1.492×	3.409	1.493×	**3.429**	**1.497×**
MVBench	3.626	1.497×	**3.649**	**1.515×**	3.645	1.497×
LongVideoBench	3.622	1.287×	**3.654**	**1.379×**	2.540	1.055×
Video-MME	3.527	1.296×	**3.567**	**1.383×**	2.674	1.132×

Qwen3-8B DFlash checkpoint (Chen et al., 2026) on HotpotQA, Qasper, and GovReport from LongBench (Bai et al., 2024). For these text-only tasks, tokens retain their original order without visual reordering. Full and Sparse use identical checkpoint weights and inputs. Table 4 shows improvements in both acceptance length and end-to-end speedup across all evaluated settings, with larger relative speed gains at 16K than at 8K.

We further evaluate the official Qwen3.5-9B DFlash checkpoint, which was trained with sliding-window attention in all draft layers except the final full-attention layer. We replace attention in either the final layer (last) or every layer (all). Replacing all draft attention layers maintains acceptance on VDC and MVBench, where input contexts are approximately 9–10K tokens long, but substantially reduces acceptance on the longer LongVideoBench and Video-MME inputs. This degradation likely reflects a mismatch between the local attention pattern used during training and the distant context introduced by retrieval, which becomes more pronounced as context length increases. Replacing only the full-attention layer preserves the original local attention pattern in the remaining layers and improves both acceptance and speedup across all four benchmarks (Table 5).

6 Conclusion

We presented SparseDFlash, a parameter-free sparse context attention framework for long-video speculative decoding. The framework reduces repeated context access during drafting while retaining the complete history for subsequent retrieval. Prefix-aware aggregation and locality-aware reordering improve block selection by accounting for prefix acceptance and the spatiotemporal structure of video. Experiments with Qwen3-VL-8B and Qwen3.5-9B across four video benchmarks demonstrate improved average end-to-end speedup over full-attention drafting using identical checkpoint weights. Ablations further show that selective retrieval can improve draft–target agreement even beyond full attention. The method also applies to full-attention layers in pre-trained block-diffusion drafters without additional training. These results demonstrate that selecting context at inference time can improve both draft acceptance and decoding speed without updating the drafter's parameters.

DECLARATION OF GENERATIVE AI USE

Generative AI assistance was limited to language editing to enhance the manuscript's clarity, grammatical accuracy, and phrasing. It played no role in developing the research methodology, formulating hypotheses or theoretical claims, planning experiments, interpreting findings, or drawing scientific conclusions. The authors reviewed and approved all revisions and accept full responsibility for the final manuscript.

REFERENCES

Marianne Arriola, Aaron Kerem Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations (ICLR)*, pp. 50726–50753, 2025.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, pp. 3119–3137. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.172.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *International Conference on Machine Learning (ICML)*, pp. 5209–5235, 2024.

Jakub Červený. gilbert. GitHub repository, n.d. URL `https://github.com/jakubcerveny/gilbert`. Accessed September 23, 2026.

Wenhao Chai, Enxin Song, Yilun Du, Chenlin Meng, Vashisht Madhavan, Omer Bar-Tal, Jenq-Neng Hwang, Saining Xie, and Christopher D. Manning. AuroraCap: Efficient, performant video detailed captioning and a new benchmark. In *International Conference on Learning Representations (ICLR)*, pp. 16845–16889, 2025.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *International Conference on Machine Learning (ICML)*, 2026.

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 *European Conference on Computer Vision (ECCV)*, pp. 19–35, 2024. doi: 10.1007/978-3-031-73004-7_2.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, pp. 35549–35562, 2024.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24108–24118, 2025.

Mugilan Ganesan, Shane Segal, Ankur Aggarwal, Nish Sinnadurai, Sean Lie, and Vithursan Thangarasa. MASSV: Multimodal adaptation and self-data distillation for speculative decoding of vision-language models. *arXiv preprint arXiv:2505.10526*, 2025.

Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, and Mao Yang. SeerAttention: Self-distilled attention gating for efficient long-context prefilling. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 55846–55869, 2025. doi: 10.52202/085713-1869.

David Hilbert. Ueber die stetige Abbildung einer Linie auf ein Flächenstück. *Mathematische Annalen*, 38:459–460, 1891. doi: 10.1007/BF01199431.

Jianuo Huang, Yaojie Zhang, Qituan Zhang, Hao Lin, Hanlin Xu, and Linfeng Zhang. Domino: Decoupling causal modeling from autoregressive drafting in speculative decoding. *arXiv preprint arXiv:2605.29707*, 2026.

Xiaohu Huang, Hao Zhou, and Kai Han. PruneVid: Visual token pruning for efficient video large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19959–19973. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.1024.

Yicheng Ji, Jun Zhang, Heming Xia, Jinpeng Chen, Lidan Shou, Gang Chen, and Huan Li. SpecVLM: Enhancing speculative decoding of video LLMs via verifier-guided token pruning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7205–7219. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.366.

Yicheng Ji, Jun Zhang, Jinpeng Chen, Cong Wang, Lidan Shou, Gang Chen, and Huan Li. See the forest for the trees: Loosely speculative decoding via visual-semantic guidance for efficient inference of video LLMs. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, pp. 23707–23726. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1087.

Jialiang Kang, Han Shu, Wenshuo Li, Yingjie Zhai, and Xinghao Chen. ViSpec: Accelerating vision-language models with vision-aware speculative decoding. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 115511–115532, 2025. doi: 10.52202/085713-3852.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning (ICML)*, pp. 19274–19286, 2023.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, Limin Wang, and Yu Qiao. MVBench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22195–22206, 2024a.

Shenggui Li, Chao Wang, Yikai Zhu, Yubo Wang, Fan Yin, Shuai Shi, Yefei Chen, Xiaomin Dong, Qiaoling Chen, Jin Pan, Ji Li, Laixin Xie, Yineng Zhang, Lei Yu, Yonggang Wen, Ivor Tsang, and Tianwei Zhang. SpecForge: A flexible and efficient open-source training framework for speculative decoding. *arXiv preprint arXiv:2603.18567*, 2026.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning (ICML)*, pp. 28935–28948, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 136737–136756, 2025. doi: 10.52202/085713-4562.

Luxi Lin, Zhihang Lin, Zhanpeng Zeng, and Rongrong Ji. Speculative decoding reimagined for multimodal large language models. *arXiv preprint arXiv:2505.14260*, 2025.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 17790–17815, 2025. doi: 10.52202/085713-0602.

Kele Shao, Keda Tao, Can Qin, Haoxuan You, Yang Sui, and Huan Wang. HoliTom: Holistic token merging for fast video large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 135547–135570, 2025. doi: 10.52202/085713-4525.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *International Conference on Machine Learning (ICML)*, pp. 47901–47911, 2024.

Keda Tao, Can Qin, Haoxuan You, Yang Sui, and Huan Wang. DyCoke: Dynamic compression of tokens for fast video large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18992–19001, 2025.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. LongVideoBench: A benchmark for long-context interleaved video-language understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 28828–28857, 2024. doi: 10.52202/079017-0907. Datasets and Benchmarks Track.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 119638–119661, 2024a. doi: 10.52202/079017-3801.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations (ICLR)*, pp. 21875–21895, 2024b.

Zhinan Xie, Peisong Wang, Shuang Qiu, and Jian Cheng. HiViS: Hiding visual tokens from the drafter for speculative decoding in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 8952–8961, 2026.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. VisionZip: Longer is better but not necessary in vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19792–19802, 2025.

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 *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, pp. 23078–23097. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1126.

Libo Zhang, Zhaoning Zhang, Wangyang Hong, Peng Qiao, and Dongsheng Li. Sparrow: Text-anchored window attention with visual-semantic glimpsing for speculative decoding in video LLMs. *arXiv preprint arXiv:2602.15318*, 2026.

Ruohong Zhang, Liangke Gui, Zhiqing Sun, Yihao Feng, Keyang Xu, Yuanhan Zhang, Di Fu, Chunyuan Li, Alexander G Hauptmann, Yonatan Bisk, and Yiming Yang. Direct preference optimization of video large multimodal models from language model reward. In *Proceedings of the Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Volume 1: Long Papers*, pp. 694–717. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.naacl-long.30.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. LLaVA-Video: Video instruction tuning with synthetic data. *Transactions on Machine Learning Research*, 2025b.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 34661–34710, 2023. doi: 10.52202/075280-1506.

Weilin Zhao, Zihan Zhou, Zhou Su, Chaojun Xiao, Yuxuan Li, Yanghao Li, Yudi Zhang, Weilun Zhao, Zhen Li, Yuxiang Huang, Ao Sun, Xu Han, and Zhiyuan Liu. InfLLM-V2: Dense-sparse switchable attention for seamless short-to-long adaptation. In *International Conference on Learning Representations (ICLR)*, pp. 69374–69390, 2026.

A SPARSE ATTENTION KERNEL

We implement draft attention with a fused Triton kernel that reads selected keys and values directly from the KV cache, without materializing an intermediate gathered KV buffer. Each program processes one query head in two passes: the window pass covers the recent context and current draft block, and the retrieved pass processes the router-selected blocks. The initial blocks are included in the selected list through forced selection by the router. Query heads within a GQA group share the same selected blocks.

Both passes maintain a shared online-softmax state following FlashAttention-2 (Dao, 2024). The running maximum, normalization sum, and output accumulator are carried across the two passes, with normalization performed only after the final tile. Entries appearing in both passes are processed twice; the kernel performs no additional deduplication. For retrieval blocks of width c, each tile of width B_N packs $P_B = B_N/c$ selected blocks, where B_N is a multiple of c. Inputs are loaded in BF16, while softmax statistics and output accumulation use FP32.

Visual reordering is implemented through an index map computed once during prefill and reused throughout decoding. Both passes apply this map when resolving logical token indices to KV locations; the cache itself is not physically reordered. All block-index, permutation, and KV loads are masked to exclude invalid entries.

Algorithm 1 describes the two passes, and Algorithm 2 gives their shared attention update. Here, T includes the current draft block, w_0 denotes the window start, and G is the number of query heads per KV head. For Q_M query positions and head dimension D, $m, \ell \in \mathbb{R}^{Q_M}$ and $A \in \mathbb{R}^{Q_M \times D}$. The window is nonempty ($w_0 < T$), so the first tile establishes finite running maxima before subsequent tiles are processed. MASKEDLOAD reads only valid entries and returns zero for masked entries. In ATTEND, Q_h, $\mathbf{K}$, $\mathbf{V}$, and s are inherited from the calling program.

Algorithm 1 Sparse attention for one draft block

Require: Queries Q, KV cache $\mathbf{K}, \mathbf{V}$, selected block lists $\mathcal{B}_g$, optional index map π
Require: Window start w_0, KV length T, block width c, tile width B_N, GQA ratio G, softmax scale s

$P_B \leftarrow B_N/c$; $\boldsymbol{\lambda} \leftarrow [0, B_N)$
for each query head h in parallel **do**
  $g \leftarrow \lfloor h/G \rfloor$; $Q_h \leftarrow Q[h, :, :]$
  $m \leftarrow [-\infty]^{Q_M}$; $\ell \leftarrow [0]^{Q_M}$; $A \leftarrow 0^{Q_M \times D}$
  Window pass
  for $n = w_0$ **to** $T - 1$ **step** B_N **do**
    $\mathbf{j} \leftarrow n + \boldsymbol{\lambda}$; $v \leftarrow \mathbf{j} < T$
    if π is provided **then**
      $\mathbf{j} \leftarrow$ MASKEDLOAD$(\pi, \mathbf{j}, v)$
    end if
    $(m, \ell, A) \leftarrow$ ATTEND$(g, \mathbf{j}, v; m, \ell, A)$
  end for
  Retrieved pass
  for $i = 0$ **to** $|\mathcal{B}_g| - 1$ **step** P_B **do**
    $\mathbf{t} \leftarrow i + \lfloor \boldsymbol{\lambda}/c \rfloor$; $\mathbf{r} \leftarrow \boldsymbol{\lambda} \bmod c$
    $v \leftarrow \mathbf{t} < |\mathcal{B}_g|$
    $\mathbf{b} \leftarrow$ MASKEDLOAD$(\mathcal{B}_g, \mathbf{t}, v)$
    $\mathbf{j} \leftarrow c\,\mathbf{b} + \mathbf{r}$; $v \leftarrow v \wedge (\mathbf{j} < T)$
    if π is provided **then**
      $\mathbf{j} \leftarrow$ MASKEDLOAD$(\pi, \mathbf{j}, v)$
    end if
    $(m, \ell, A) \leftarrow$ ATTEND$(g, \mathbf{j}, v; m, \ell, A)$
  end for
  $\mathrm{Out}[h, :, :] \leftarrow \mathrm{diag}(\ell)^{-1} A$
end for

Algorithm 2 ATTEND: shared online-softmax update

Require: KV head g, token indices $\mathbf{j}$, validity mask v, state (m, ℓ, A)
$K_t \leftarrow \textsc{MaskedLoad}(\mathbf{K}_g, \mathbf{j}, v)$; $V_t \leftarrow \textsc{MaskedLoad}(\mathbf{V}_g, \mathbf{j}, v)$
$S \leftarrow s\, Q_h K_t^\top$
$S \leftarrow \mathrm{where}(v, S, -\infty)$
$m' \leftarrow \max(m, \mathrm{rowmax}(S))$
$P \leftarrow \exp(S - m')$; $\alpha \leftarrow \exp(m - m')$
$\ell \leftarrow \alpha \odot \ell + \mathrm{rowsum}(P)$
$A \leftarrow \mathrm{diag}(\alpha) A + P V_t$
$m \leftarrow m'$
return (m, ℓ, A)

B RETRIEVAL BUDGET ACROSS CONTEXT LENGTHS

Tables 6 and 7 give the full grid summarised by Figure 3: Qwen3-VL-8B on the Video-MME long split, clips longer than 600 s sampled at 2 fps under a fixed 256 Mi total-pixel budget, 256 generated tokens. The three context columns are formed by capping the decoded frame count at 586, 1170 and 2340; the cap binds on all but the shortest videos, giving median prefills of 35,928, 71,660 and 143,288 tokens. The budget K counts the always-visible initial block ($k_{\text{init}} = 1$, $k_{\text{global}} = K - 1$), so $K = 1$ routes nothing and reads only the initial block and the local window w. The last column is the same checkpoint run with full attention.

Table 6: Acceptance length τ under varying local window size w and block budget K.

context	w	K=1	K=4	K=16	K=32	K=64	full
32K	256	2.379	2.479	2.536	2.540	2.565	2.409
	512	2.405	2.485	2.537	2.539	2.559	
	1024	2.430	2.498	2.539	2.544	2.564	
64K	256	2.271	2.304	2.351	2.356	2.362	2.371
	512	2.294	2.324	2.340	2.362	2.379	
	1024	2.298	2.330	2.350	2.355	2.373	
128K	256	2.267	2.307	2.335	2.345	2.350	2.332
	512	2.283	2.332	2.370	2.370	2.381	
	1024	2.286	2.331	2.349	2.356	2.369	

Table 7: Decode-only speedup under varying local window size w and block budget K.

context	w	K=1	K=4	K=16	K=32	K=64	full
32K	256	1.981×	2.062×	2.130×	2.137×	2.150×	2.030×
	512	2.047×	2.074×	2.088×	2.104×	2.109×	
	1024	2.057×	2.074×	2.112×	2.110×	2.118×	
64K	256	1.833×	1.883×	1.917×	1.915×	1.913×	1.881×
	512	1.868×	1.887×	1.890×	1.916×	1.937×	
	1024	1.892×	1.902×	1.909×	1.920×	1.933×	
128K	256	1.362×	1.382×	1.390×	1.394×	1.407×	1.264×
	512	1.367×	1.391×	1.414×	1.416×	1.427×	
	1024	1.366×	1.394×	1.401×	1.406×	1.421×	

C IMPLEMENTATION DETAILS

SparseDFlash is applied to a compatible DFlash checkpoint with its weights frozen. This section describes the preparation of the Qwen3-VL-8B and Qwen3.5-9B DFlash checkpoints used in the main experiments. For each target, DFlash and SparseDFlash use the same resulting checkpoint. The official DFlash checkpoints used in Table 4 and Table 5 were not trained with the settings described here.

Table 8: DFlash training settings. Stage 2 is initialised from the Stage-1 checkpoint.

	Stage 1	Stage 2
Data	LLaVA-Hound + ShareGPT4Video	LLaVA-Video
Maximum length	8,192	16,384
Schedule	3 epochs	6,250 steps
Learning rate	6×10^{-4}	1×10^{-4}
Batch size per device	2	2
GPUs	8×B200	8×B200

Draft architecture. The drafter configurations for Qwen3-VL-8B and Qwen3.5-9B are based on the official DFlash configurations for Qwen3-8B and Qwen3.5-9B, respectively (Chen et al., 2026). Settings not specified below follow these official configurations. Every drafter parameter is initialised randomly. The Qwen3-VL-8B and Qwen3.5-9B drafters contain 5 and 6 full-attention layers, respectively. Both use a hidden size of 4,096, an intermediate size of 12,288, 32 query heads, and 8 key/value heads with a head dimension of 128, and retain the target's vocabulary and tokenizer. Target hidden states are extracted from layers $\{3, 9, 17, 25, 33\}$ of the 36-layer Qwen3-VL-8B and $\{3, 5, 9, 13, 17, 21, 25, 29\}$ of the 32-layer Qwen3.5-9B. Both drafters use a block size of 16.

Training. The target remains frozen while the drafter is trained using the SpecForge framework (Li et al., 2026). We use the DFlash cross-entropy objective with position-dependent loss weighting and a decay scale of $\gamma = 7$. Both training stages use full context attention. Training uses AdamW with a cosine learning-rate schedule and a warm-up ratio of 0.04. Stage 1 samples 64 frames per video; Stage 2 samples at 2 fps with a maximum of 256 frames. All runs use 8 B200 GPUs with a per-device batch size of 2, giving an effective batch size of 16. Table 8 summarises the training settings.

D Results across Model Sizes

We evaluate SparseDFlash with Qwen3.5-4B, Qwen3.5-9B, and Qwen3.5-27B using their official DFlash checkpoints on the same 50 Video-MME clips and evaluation protocol as the main experiments. For each checkpoint, we replace only the final full-attention layer with sparse context attention, retaining the remaining sliding-window layers without additional training. Table 9 shows that SparseDFlash improves acceptance length, end-to-end speedup, and decode-only speedup over DFlash across all three target model sizes.

Table 9: **Results across target model sizes.** Acceptance length (τ), end-to-end speedup, and decode-only speedup on 50 Video-MME clips. DFlash and SparseDFlash use identical checkpoint weights. Both speedups are relative to autoregressive decoding.

Target	Method	τ	E2E speedup	Decode speedup
Qwen3.5-4B	DFlash	3.158	1.159×	1.240×
	SparseDFlash	**3.202**	**1.215×**	**1.342×**
Qwen3.5-9B	DFlash	3.315	1.171×	1.271×
	SparseDFlash	**3.369**	**1.237×**	**1.411×**
Qwen3.5-27B	DFlash	3.430	1.373×	1.491×
	SparseDFlash	**3.468**	**1.387×**	**1.511×**