Stateful Token Reduction for Long-Video Hybrid VLMs

Jindong Jiang* **Amala Sanjay Deshmukh** **Kateryna Chumachenko** **Karan Sapra** **Zhiding Yu** **Guilin Liu** **Andrew Tao** **Pavlo Molchanov** **Jan Kautz** **Wonmin Byeon***

NVIDIA

arXiv:2603.00198v2 [cs.CV] 1 Jul 2026

Token reduction accelerates long-video vision–language models (VLMs), but existing methods target Transformers, where reduction is treated as token pruning. We study token reduction in hybrid Mamba–Transformer VLMs and find that it is *stateful*: Mamba layers maintain a recurrent state that accumulates information from earlier tokens, allowing discarded tokens to persist, so reduction behaves more like compression than dropping. We support this view with a representation-based probing method measuring how much information from discarded tokens is retained, and analyze layer-wise sparsity and cross-layer importance stability. Our findings show importance is sparse within layers but unstable across layers, making aggressive early pruning unreliable while hybrids remain robust to later reduction. Motivated by this, we propose a hybrid-aware token reduction framework with a low-to-high progressive schedule and a unified query-conditioned importance score for attention and Mamba layers. For Mamba, excluding the position-dependent decay from the recurrence produces a stronger selection signal. Across long-video benchmarks, our method achieves 3.8–4.2× prefilling speedups at a 25% token budget while maintaining near-baseline accuracy and improving with light finetuning. Hybrid models benefit from aggressive reduction, improving both efficiency and accuracy, whereas Transformers exhibit the standard trade-off. Our method also outperforms prior baselines on the same hybrid backbone and combines effectively with visual redundancy reduction methods.

1. Introduction

Video-based vision–language models (VLMs) are moving toward long-horizon video understanding tasks. However, long-video VLMs process thousands of visual tokens, making inference expensive, especially during prefilling. Prior work shows that many tokens are redundant and can be pruned or merged, but aggressive reduction often degrades accuracy, particularly when applied early, where errors cannot be corrected in later layers [1, 2, 3]. This becomes more challenging for long videos, where token relevance varies across depth and time [4, 5, 6].

Existing token-reduction methods are primarily designed for Transformer architectures, where reduction is naturally interpreted as token pruning. However, recent VLMs increasingly adopt hybrid Mamba–Transformer architectures [7, 8], in which a substantial fraction of layers are state-space (Mamba) blocks. These architectures propagate information differently, but their implications for token reduction remain underexplored.

In this work, we study token reduction in hybrid VLMs and find it exhibits *stateful* behavior. In contrast to Transformers, where removed tokens no longer influence subsequent layers, Mamba layers maintain a recurrent state that accumulates information from earlier tokens and carries it across layers. As a result, information from discarded tokens can persist in the state, and reduction behaves more like compression than dropping.

We show a representation-based probe measuring how much information from discarded tokens is preserved in retained tokens, providing empirical evidence for the stateful behavior of hybrid models. This observation has implications for reduction design. Complementing this, we analyze layer-wise sparsity and cross-layer importance stability, and find that while importance is sparse within each layer, it is unstable across layers, making aggressive early pruning unreliable. At the same time, hybrid models show stronger tolerance to reduction, consistent with information being absorbed into the recurrent state. These findings motivate a progressive reduction strategy that preserves more tokens in early layers and prunes more aggressively later.

Based on this analysis, we propose a hybrid-aware token reduction framework. We adopt a low-to-high progressive schedule and introduce a unified query-conditioned importance score for both attention and Mamba layers. For attention layers, importance is derived from text-to-vision attention. For Mamba layers, we derive an attention-like score from the selective state-space recurrence and show that the

* : Equal contribution

© 2026 NVIDIA. All rights reserved.

content-alignment term alone is a more effective signal than the full recurrent weight, which includes position-dependent decay.

Experiments on long-video benchmarks (VideoMME [9], LongVideoBench [10], and LVBench [11]) show that the proposed method achieves substantial prefilling and end-to-end speedups. Notably, hybrid models maintain or improve accuracy under aggressive token reduction, whereas Transformer models exhibit the standard speed–accuracy trade-off. Ours also outperforms existing token-reduction baselines on the same hybrid backbone and combines effectively with visual redundancy reduction methods.

Contributions. (i) We study token reduction in Mamba–Transformer hybrid VLMs and identify its stateful behavior, supported by a representation-based probing analysis. (ii) We propose a hybrid-aware token reduction framework with a progressive schedule and a unified query-conditioned scoring method across attention and Mamba layers, and show that excluding cumulative decay is critical for Mamba-based selection. (iii) We demonstrate strong speed–quality trade-offs on long-video benchmarks, with hybrid models benefiting from token reduction more than Transformer baselines.

The rest of the paper is organized as follows. Section 2 presents the empirical observations that motivate our design. Section 3 describes the scoring rule and the progressive schedule. Section 4 reports the main results, ablations, and efficiency analysis. A broader discussion of related work is provided in Section 5 and Section A.8 (Appendix).

2. Observations

We compare a Transformer VLM (Qwen3-VL) and a hybrid Mamba–Transformer VLM (Nemotron-Nano-V2 VL) along three properties that motivate the method in Section 3: *(i) stateful compression*, *(ii) layer-wise sparsity*, and *(iii) cross-layer importance stability*. We use the query-conditioned token-importance score whose precise definition is deferred to Section 3.1.

2.1. Stateful Compression

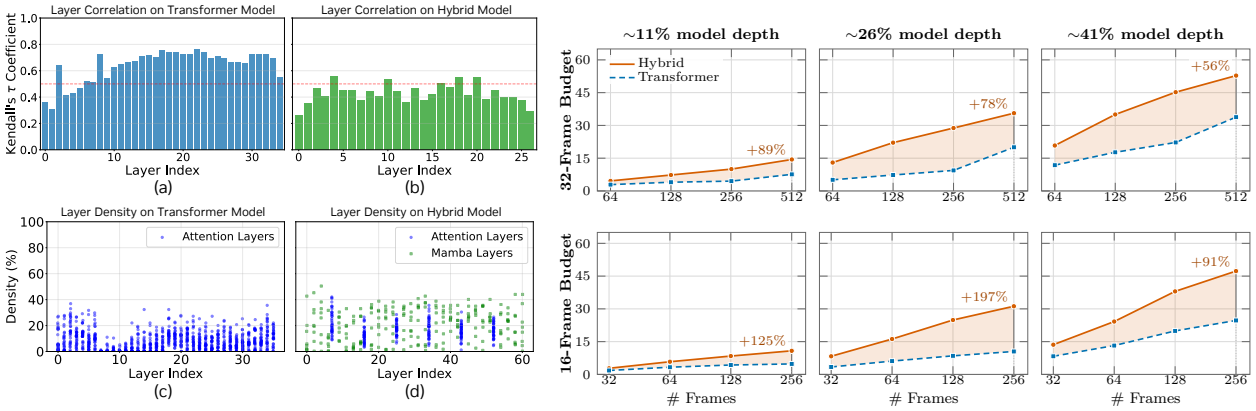
We hypothesize that consistency and reduction tolerance are driven by different mechanisms. In standard Transformers, attention is memoryless, so when the tokens are removed, their information is effectively *dropped*. Once pruned, these tokens cannot influence later layers [12]. Hybrid models behave differently. They interleave attention with Mamba layers that maintain a recurrent state $\boldsymbol{S}_t$. This allows token reduction to act more like *compression* rather than dropping. Even if a token is discarded, its information can persist in the state and be propagated to the remaining tokens.

Probing Stateful Compression. We test this idea using a representation probe. First, we run a full forward pass and select the top-K tokens based on their importance at layer ℓ. Then, we run a second forward pass where only these K tokens are kept from layer 1 onward. At layer ℓ, we compare the hidden states of the retained tokens between the two passes using angular distance. If the model simply drops tokens, the two representations should be very similar. In contrast, a larger angular distance indicates that earlier layers have already absorbed information from the discarded tokens into the retained ones. Figure 1ii shows results across two retention budgets equivalent to 32 and 16 frames, three depths (∼11%, ∼26%, ∼41%), and up to 512 frames. The hybrid model consistently shows larger angular distances than the Transformer (e.g., 45.2° vs. 22.2° at 41% depth, 32-frame budget, 256 frames). Moreover, this gap increases with both depth and number of frames, supporting the idea that hybrid models perform stateful compression rather than simple token dropping.

2.2. Layer Sparsity Statistics

Figure 1i (bottom) plots the fraction of tokens that account for 80% of the importance mass at each layer; each dot is one attention head (blue) or one Mamba group (green). The majority fall below 40% in both architectures, with average density of 10.9% in the Transformer and 30.6% in the hybrid, suggesting that token reduction is feasible in principle for both. Transformer attention is sparser than Mamba on average, and within the hybrid, attention is also sparser than Mamba groups.

Sparsity tells us *which* tokens matter at a given layer, but not whether the *same* tokens remain important across layers. Figure 1i (top) reports Kendall's τ between consecutive layers' importance rankings; values below 0.5 mean more than 25% of token pairs reverse their relative ordering. Early-layer correlations are unstable in both architectures: the Transformer stabilizes only after roughly one-third of network depth, while the hybrid remains consistently low and on average lower than the Transformer's. The implication is that single-shot pruning at the very first layer is risky and a depth-aware schedule is preferable in either case.



(i) Layer-wise attention density and cross-layer rank stability of token importance.

(ii) Stateful compression probing via angular distance of the hidden representations between full and the top-K retained tokens.

Figure 1 | **Token importance structure and compression behavior in Hybrid vs Transformer models.** **(i) Top:** Kendall's τ [13] between the importance rankings of consecutive layers, for the Transformer (a) and the hybrid (b); values around 0.5 and below indicate low rank consistency. **Bottom:** layer-wise density (%) of importance scores for the Transformer (c) and the hybrid (d), where each dot is one attention head (blue) or one Mamba group (green); lower density indicates higher sparsity. **(ii)** Angular distance between hidden states of retained tokens, quantifies how much information from discarded tokens is absorbed. Larger distances indicate more information absorbed from the discarded tokens, and small distance means the representations are almost unchanged, implying little additional information was incorporated. The Hybrid model consistently shows greater compression, with the gap widening at deeper layers and higher frame counts.

Takeaways. The stronger stateful-compression effect in hybrids both *permits* and *rewards* preserving more tokens in early layers, so that the recurrent state can absorb information before later layers prune more aggressively. Together with (ii) and (iii), this motivates the low-to-high progressive schedule in Section 3, which we expect to benefit hybrids in particular.

3. Method

3.1. Query-Conditioned Token Importance

Consider a multimodal sequence with M text (query) tokens and N visual tokens. For long videos, N can exceed 10K, creating computational bottlenecks, while M is typically much smaller (e.g., ∼100). Let $\boldsymbol{u}_1^{(\ell)}, \ldots, \boldsymbol{u}_M^{(\ell)}$ and $\boldsymbol{v}_1^{(\ell)}, \ldots, \boldsymbol{v}_N^{(\ell)}$ denote the hidden states of text and visual tokens at layer ℓ. Our goal is to define a *query-conditioned importance score* $s_i^{(\ell)} \in \mathbb{R}$ for each visual token to measure relevance to the query at layer ℓ and enable ranking and pruning: given budget $K < N$, we retain the top-K tokens by importance.

Attention Layers. For attention layers, importance follows naturally from the attention mechanism. Let $\boldsymbol{q}_m^{(h)}$ and $\boldsymbol{k}_i^{(h)}$ denote the query and key projections of $\boldsymbol{u}_m^{(\ell)}$ and $\boldsymbol{v}_i^{(\ell)}$ at head h (we omit the layer index for brevity). We compute the softmax-normalized attention weights from text to visual tokens and aggregate: $s_i^{(\ell,\text{attn})} = \frac{1}{MH} \sum_{m,h} \text{softmax}_i \left(\frac{\boldsymbol{q}_m^{(h)} \cdot \boldsymbol{k}_i^{(h)}}{\sqrt{d_h}} \right)$, where H is the number of heads, and d_h is the head dimension.

State-Space (Mamba) Layers. For Mamba layers, we derive an implicit attention proxy from the selective state-space recurrence [14]. Each head in Mamba operates independently on input $\boldsymbol{x}_t \in \mathbb{R}^p$ (where p is the head dimension) with hidden state $\boldsymbol{S}_t \in \mathbb{R}^{p \times n}$ (where n is the state dimension). The state evolves as $\boldsymbol{S}_t = \bar{\boldsymbol{A}}_t \boldsymbol{S}_{t-1} + \boldsymbol{x}_t \bar{\boldsymbol{b}}_t^\top$, with output $\boldsymbol{y}_t = \boldsymbol{S}_t \boldsymbol{c}_t$, where $\bar{\boldsymbol{A}}_t$ is the discretized state transition matrix with entries in $(0, 1)$ acting as a decay factor; $\bar{\boldsymbol{b}}_t = \Delta_t \boldsymbol{b}_t \in \mathbb{R}^n$ is the effective input projection combining the discretization step $\Delta_t > 0$ with the input-dependent projection $\boldsymbol{b}_t$; and $\boldsymbol{c}_t \in \mathbb{R}^n$ is the output projection. Mamba groups heads such that $\boldsymbol{b}_t$ and $\boldsymbol{c}_t$ are shared within each group. Unrolling the recurrence reveals an attention-like structure $\boldsymbol{y}_t = \sum_{j=1}^t w_{t,j} \boldsymbol{x}_j$, where $w_{t,j}$ is a scalar weight quantifying the contribution of token j to the output at position t:

$$w_{t,j} = \left(\prod_{u=j+1}^{t} \bar{\boldsymbol{A}}_u \right) \bar{\boldsymbol{b}}_j^\top \boldsymbol{c}_t. \tag{1}$$

This weight depends on two factors: (1) a *content alignment* $\bar{\boldsymbol{b}}_j^\top \boldsymbol{c}_t$, where $\bar{\boldsymbol{b}}_j$ and $\boldsymbol{c}_t$ act analogously to keys and queries [15, 16], and (2) a *cumulative decay*

$\prod_u \bar{\boldsymbol{A}}_u$ that monotonically attenuates older positions. We provide a detailed derivation of this attention connection in Section A.2.

Why the Full Mamba Dynamic Fails. Given that Equation (1) is the exact contribution of visual token j to the output at text position t, the most natural "attention-like" importance score is simply $|w_{t,j}|$, in direct analogy to softmax attention scores. We refer to this score as the *Full Mamba Dynamic* because it preserves the full selective-scan dynamics, content alignment together with positional decay. However, we find that this choice performs substantially worse than what we propose below. Concretely, applying the Full Mamba Dynamic at ∼25% token budget drops the average accuracy by up to 8.1 points relative to the no-reduction baseline, depending on which Mamba layers participate in selection (see Table 4 in Section 4). The cause is structural: the diagonal entries of $\bar{\boldsymbol{A}}_u$ in Mamba layers are bounded in $(0, 1)$ and their cumulative product $\prod_u \bar{\boldsymbol{A}}_u$ can decay sharply with distance, so $w_{t,j}$ assigns near-zero weight to early visual tokens in some layers regardless of how relevant their content is to the query. We visualize this effect in some layers in Figure 4 (Section A.2.3).

Decoupling Content from Position. This failure mode suggests separating the *data-dependent* content alignment from the *position-dependent* decay, and using only the former for token selection. We refer to the resulting score as the *Decoupled Content Alignment*, defined for Mamba layers as

$$s_i^{(\ell,\mathrm{ssm})} = \frac{1}{MG} \sum_{m,g} \left| \bar{\boldsymbol{b}}_i^{(g)\top} \boldsymbol{c}_m^{(g)} \right|, \tag{2}$$

averaging over M query (text) positions and G Mamba groups, where $\bar{\boldsymbol{b}}_i$ and $\boldsymbol{c}_m$ act as key and query projections whose dot product measures alignment between visual token i and text token m, with the gating factor Δ_i absorbed into $\bar{\boldsymbol{b}}_i$. Despite its simplicity, this decomposition is, to our knowledge, the first explicit identification of which component of the Mamba recurrence is suitable for content-based token scoring and which component should be excluded; we validate this empirically against the Full Mamba Dynamic in Table 4.

Token Selection. Given scores $\{s_i\}_{i=1}^N$, we select the top-K tokens while preserving temporal order. The budget K may vary across layers according to a reduction schedule, which we describe next, motivated by the observations in Section 2.

3.2. Reduction Scheduling

We investigate how different reduction schedules interact with hybrid architectures, exploring a design space along two key dimensions: (1) *where* to apply reduction, i.e., which layer types; and (2) *when* to reduce, i.e., early versus progressive. Figure 2i illustrates the schedules we consider.

Layer Type Selection. Hybrid models interleave attention and Mamba layers (see Figure 2i), raising the question of where to apply reduction. We consider three options: *attention-only*, where only attention layers compute importance and prune tokens; *attention + Mamba*, which inserts one or two Mamba reductions between attention layers; and *all layers*, where reduction is applied throughout the model. Attention-only is simpler and leverages attention sparsity, while including Mamba enables finer-grained pruning but requires computing the implicit importance scores in Equation (2).

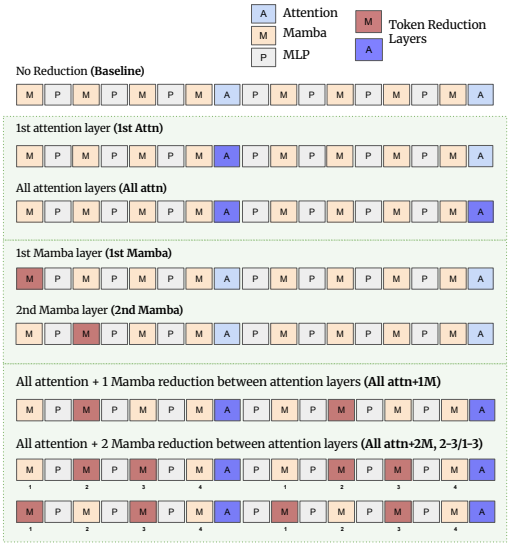
Early versus Progressive Reduction. A second design choice is whether to reduce tokens early or progressively. *Early reduction* prunes tokens once at the LLM input using early-layer importance scores [17], while *progressive reduction* retains more tokens in shallow layers and gradually reduces the budget with depth. Motivated by our analysis (Section 2), i.e., early importance is unreliable, and Mamba layers can absorb information from discarded tokens, we adopt a low-to-high progressive schedule. We parameterize the retention ratio $r^{(\ell)}$ as a monotonic decay from r_{start} to r_{end}. We consider (1) *step decay*, which reduces tokens in stages, and (2) *sigmoid decay* [18], which enables smooth, late-stage reduction via a midpoint-controlled curve, defined as $r^{(\ell)} = r_{\text{end}} + (r_{\text{start}} - r_{\text{end}}) \cdot \sigma(-k(\ell/L - x_0))$, where L is the number of layers, k controls steepness, and x_0 the midpoint. We fix $k = 20$ and vary x_0 to control reduction position. Across target budgets (25%, 35%, 50%), preserving more tokens in early layers matters more than the specific decay form. The reduction curve examples are shown in Figure 2ii.

We evaluate these scheduling and layer-type tradeoffs empirically in Section 4.

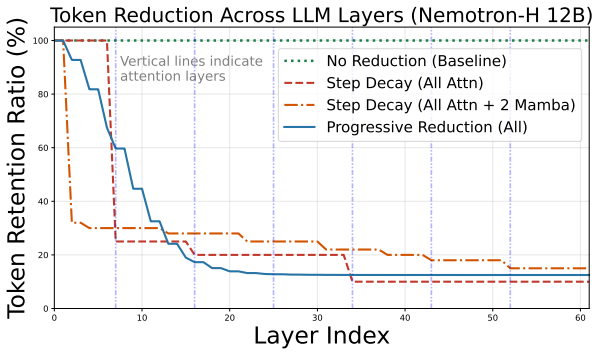
4. Experiments

4.1. Experimental Setup

We use SigLIP2 [19] (384×384) to produce 576 patch tokens, pooled into 144 tokens/frame via an MLP projector. We compare a dense Transformer



(i) Hybrid Token Reduction Patterns



(ii) Hybrid reduction schedules

Figure 2 | **Token-reduction patterns and schedules used for the hybrid model. (i)** Layer types (Mamba/MLP/Attention) and reduction locations for baseline, attention-only, Mamba-only, and hybrid schedules (`All attn+1M`/`All attn+2M`). **(ii)** We visualize how the token retention ratio changes with depth: (1) no reduction, (2) step decay at attention layers, (3) step decay at attention + Mamba layers, and (4) progressive low-to-high reduction applied throughout the network. Vertical dotted lines mark attention layer locations.

(Qwen3-VL 8B [20]) and a Mamba–Transformer hybrid (Nemotron-Nano-V2 VL 12B [8]) under the same vision encoder and training pipeline. Training has four stages—alignment, SFT, long-context finetuning, and token-reduction finetuning—with frames increasing from 32→64→128 (96 for the no-reduction baseline). We align using 95K image–text pairs [21], perform SFT on ∼12.5M mixed data, finetune on LLaVA-Video [22], and finally train with token reduction on EAGLE-Video-110K [23]. We evaluate on three long-context video benchmarks: VideoMME [9], LongVideoBench (LongVB) [10], and LVBench [11]. VideoMME is reported without subtitles to isolate visual reasoning, while LongVB and LVBench focus on increasingly long videos (up to ∼1–2 hours) for long-horizon understanding. Full architectural, training, and evaluation benchmark details are provided in the appendix A.3.

Reduction Schedules. To systematically evaluate the design choices discussed in Section 3.2, we first test target token budgets of approximately 25%, 35%, and 50% under an *attention-only* reduction setting, which isolates the effect of budget on accuracy and latency (Table 3). We then fix the budget to 25% and compare different layer subsets, i.e., *attention-only*, *Mamba-only*, and *attention+Mamba*, to study how reduction interacts with layer type (Table 1). For *Mamba-only*, we additionally test early-layer-only reduction (1st vs. 2nd Mamba) to examine the sparsity/stability effects observed by the analysis in Section 2. For *attention-only*, we compare pruning at a single attention layer (1st attention) vs. all attention layers. Finally, to study reduction granularity, we evaluate intermediate hybrid configurations (*attention + 1 or 2 Mamba*) that interpolate between attention-only and all-layer reduction. We visualize the layer-wise schedules and hybrid layer-patterns used in our experiments in Figure 2ii and Figure 2i. We report all schedule parameters in Section A.4.

4.2. Results

Table 1 summarizes reduction schedules on hybrid (Nemotron-Nano-V2 VL 12B) across VideoMME, LongVB, and LVBench with TTFT and reduction overheads. Under test-time reduction (25% budget), accuracy is largely preserved but depends on where reduction is applied: reducing first attention layer lowers avg. to 60.74 (−1.23), while adding Mamba reduction (`All attn+1M`) recovers long-horizon performance (LVBench 53.07, +1.68) and keeps avg. near baseline (61.88, −0.09).

Train-time reduction is consistently stronger: reducing all attention layers (`All attn`) improves all benchmarks (Avg 63.27, +1.30), and reducing all layers (`All`) performs best (Avg 63.34, +1.37). Overall, query-aware reduction improves both efficiency and long-context reasoning.

Token Reduction	Reduction Layers	# Red. Layers	VideoMME (w/o sub) (1∼60m)	LongVB (8s∼60m)	LVBench (30m∼2h)	Avg	TTFT (s)	Reduction Overhead (s)
Baseline	N/A	0	69.22	65.30	51.39	61.97	2.26	0
Test-Time Reduction								
Attn only	1st Attn	1	68.93 (-0.29)	63.05 (-2.25)	50.23 (-1.16)	60.74 (-1.23)	1.12 (×4.2)	0.84
	All attn	6	69.22 (+0.00)	64.62 (-0.68)	51.26 (-0.13)	61.70 (-0.27)	1.14 (×4.1)	0.90
Mamba only	1st Mamba	1	67.70 (-1.52)	62.60 (-2.70)	48.81 (-2.58)	59.70 (-2.27)	1.20 (×3.9)	0.94
	2nd Mamba	1	67.93 (-1.29)	62.23 (-3.07)	50.94 (-0.45)	60.37 (-1.60)	1.22 (×3.8)	0.97
Mamba+Attn	All attn+1M	13	69.22 (+0.00)	63.35 (-1.95)	**53.07 (+1.68)**	61.88 (-0.09)	1.20 (×3.9)	0.98
	All attn+2M	20	**69.26 (+0.04)**	63.05 (-2.25)	52.10 (+0.71)	61.47 (-0.50)	1.20 (×3.9)	0.98
	All	32	68.85 (-0.37)	**64.92 (-0.38)**	52.16 (+0.77)	**61.98 (+0.01)**	1.21 (×3.8)	1.00
Train-Time Reduction								
Attn only	1st attn	1	68.74 (-0.48)	64.92 (-0.38)	52.87 (+1.48)	62.18 (+0.21)	1.12 (×4.2)	0.84
	All attn	6	69.59 (+0.37)	**66.12 (+0.82)**	54.10 (+2.71)	63.27 (+1.30)	1.14 (×4.1)	0.90
Mamba only	1st Mamba	1	68.04 (-1.18)	64.10 (-1.20)	51.00 (-0.39)	61.05 (-0.92)	1.20 (×3.9)	0.94
	2nd Mamba	1	67.07 (-2.15)	63.50 (-1.80)	52.42 (+1.03)	61.00 (-0.97)	1.22 (×3.8)	0.97
Mamba+Attn	All attn+1M	13	69.00 (-0.22)	63.35 (-1.95)	52.62 (+1.23)	61.66 (-0.31)	1.20 (×3.9)	0.98
	All attn+2M	20	69.11 (-0.11)	63.80 (-1.50)	52.81 (+1.42)	61.91 (-0.06)	1.20 (×3.9)	0.98
	All	32	**69.70 (+0.48)**	66.04 (+0.74)	**54.29 (+2.90)**	**63.34 (+1.37)**	1.21 (×3.8)	1.00

Table 1 | **Token reduction at 25% compression for Nemotron-Nano-V2 VL 12B with various patterns.** We compare test-time (no finetuning) and train-time reduction (finetuned with reduction). "Reduction Layers" indicates where reduction is applied and "# Red. Layers" counts them; (·) denotes change from baseline; TTFT speedups are relative to baseline. Patterns: `1st Attn/1st Mamba`, `All Attn`, `All Attn+1M/2M`, and `All`; see Figure 2i. LLM-stage TTFT (including overhead) and the reduction overheads are measured on a single NVIDIA A100 with 256-frame input.

Method	VideoMME	LongVB	LVBench	Avg	Latency
Baseline	69.22	65.30	51.39	61.93	5.57
FastVID [24]	69.59	63.65	51.13	61.46	3.25
CDPruner [25]	69.30	53.78	36.86	53.31	2.70
FlexSelect [6] (55% budget)	69.33	65.30	52.61	62.41	3.40
FlexSelect-Lite [6] †	69.22	65.30	51.39	61.97	2.07
Ours	68.85	64.92	52.16	61.98	2.13
Ours †	69.70	**66.04**	54.29	63.34	2.13
FastVID [24] + Ours	**70.41**	65.97	**54.74**	**63.71**	3.25

Table 2 | **Comparison with token-reduction baselines on the same Nemotron-Nano-V2 VL 12B backbone.** All methods use a 25% token budget, except FlexSelect (55%). † denotes trained variants. Ours is the strongest standalone at 25%, and combining with FastVID yields further gains, highlighting complementarity between redundancy and query-conditioned reduction.

Comparison with Token-Reduction Baselines. Table 2 compares our method with prior token-reduction approaches on Nemotron-Nano-V2 VL 12B, including CDPruner [25], a training-free query-conditioned method that selects visually diverse and instruction-relevant tokens, FastVID [24], a query-agnostic method that reduces visual redundancy through temporal segmentation and density-based spatiotemporal pruning, and FlexSelect [6] that performs query-conditioned tokens selection from a reference layer (or via a learned selector in FlexSelect-Lite). CDPruner degrades significantly in our hybrid long-video setting, especially on LVBench, while FastVID is more competitive but remains below our hybrid-aware method.

In FlexSelect, following the original setup, we use layer 25, which limits compression to 55% since earlier layers process all tokens. Even at this higher budget, FlexSelect underperforms our train-time result at 25%. FlexSelect-Lite reaches 25% compression but only matches the baseline, while our train-time method exceeds it by +1.37. Our train-time variant achieves the best overall result at 25%.

Importantly, our method is complementary to query-agnostic token-reduction approaches, which remove visual redundancy independently of the query, while ours performs query-conditioned selection within the backbone. For example, combining our method with FastVID yields further gains over either alone, achieving the best overall performance (63.71) without training. This suggests that our method can be effectively composed with other redundancy reduction methods.

Compression Effects Across Architectures. Table 3 varies token budgets (50%–25%) for hybrid and Transformer models with fixed six-layer attention reduction and brief finetuning. The architectures show distinct speed–quality trade-offs. The hybrid

Token Reduction	Comp. Rate (%)	VideoMME (1∼60m)	LongVB (8s∼60m)	LVBench (30m∼2h)	Avg	TTFT(s)
Nemotron-Nano-V2 VLM 12B						
Baseline	100	69.22	65.30	51.39	61.97	4.65
All attention layers	50.1	70.00 (+0.78)	65.74 (+0.44)	54.42 (+3.03)	63.39 (+1.42)	2.21 (×2.1)
	34.7	70.26 (+1.04)	66.04 (+0.74)	54.29 (+2.9)	63.53 (+1.56)	1.53 (×3)
	25.2	69.59 (+0.37)	66.12 (+0.82)	54.10 (+2.71)	63.27 (+1.3)	1.14 (×4.1)
Qwen3-VL 8B						
Baseline	100	69.89	66.64	50.94	62.49	4.78
All attention layers	50.1	70.52 (+0.63)	66.79 (+0.15)	46.93 (-4.01)	61.41 (-1.08)	2.37 (×2)
	34.7	68.93 (-0.96)	66.12 (-0.52)	45.13 (-5.81)	60.06 (-2.43)	1.61 (×3)
	25.1	68.41 (-1.48)	64.62 (-2.02)	43.19 (-7.75)	58.74 (-3.75)	1.18 (×4.1)

Table 3 | **Varying compression rates across architectures.** We apply trained reduction at all attention layers (6 layers) for Nemotron-Nano-V2 VL and Qwen3-VL. "Comp. Rate" is the remaining token ratio; (·) denotes change from baseline. TTFT is measured on an NVIDIA A100 at 256 frames.

model (Nemotron-Nano-V2) improves performance across all budgets (Avg +1.3–+1.56) while reducing TTFT from 4.65 s to 2.21–1.14 s (2.1–4.1×). In contrast, the Transformer (Qwen3-VL) improves only at 50% and degrades at lower budgets. This supports our analysis (Section 2) that early compression in Transformers causes unrecoverable information loss. Trends are consistent across schedules (Section 4.5). Overall, reduction improves both speed and quality in hybrids, while Transformers exhibit a standard trade-off.

4.3. Efficiency Analysis

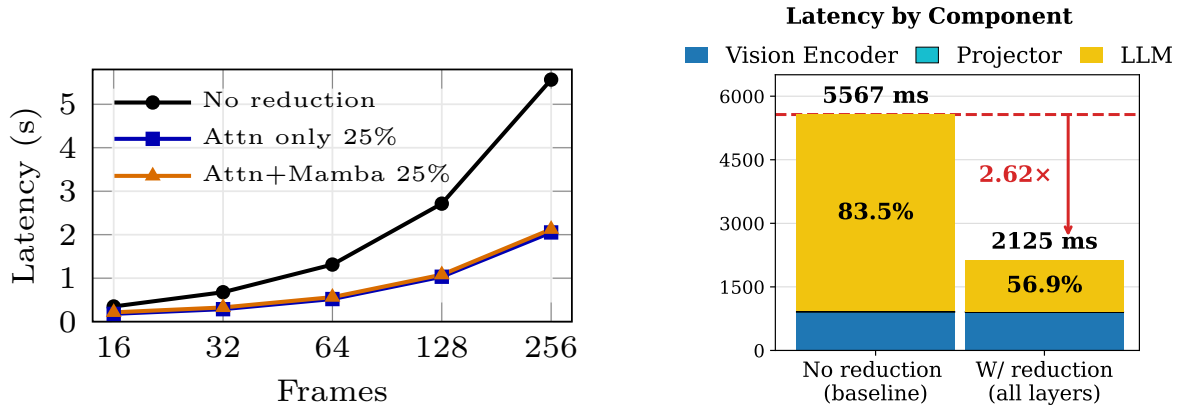
Latency Analysis. Table 1 reports TTFT and reduction overhead for Nemotron-Nano-V2 on an A100 with 256-frame input. At ∼25% token budget, TTFT drops from 4.65 s to 1.12–1.22 s with 0.84–1.00 s overhead, achieving 3.8–4.2× speedups. Adding Mamba reduction slightly increases overhead but maintains speedups. Including vision encoder and projector costs, Figure 3i shows end-to-end latency decreases from 5.567 s to 2.125 s (2.62×) at 256 frames, with the gap widening for longer videos as LLM cost scales faster. Without reduction, latency grows steeply and reaches out-of-memory (OOM) at 512 frames (see Figure 5i in Appendix). Figure 3ii further decomposes latency into vision encoder, projector, and LLM stages, showing the baseline is dominated by the LLM (83.5%). With all-layer reduction, LLM latency drops to 56.9%, confirming that long-video inference is primarily bounded by LLM compute.

Training Efficiency. Token reduction also improves training efficiency. As shown in Table 6, with 64-frame inputs, 25% reduction yields 1.48× faster training, 1.45× higher throughput, and 8.7% lower peak GPU memory. More importantly, under the same A100 80GB memory budget, token reduction supports 128-frame finetuning, whereas the no-reduction baseline is limited to 96 frames. Thus, token reduction not only accelerates training but also makes longer-video training more feasible under fixed hardware constraints.

4.4. Ablations

Pattern	Score	Video MME	LongVB	LVBench	Avg
Baseline	N/A	69.22	65.30	51.39	61.97
All attn + 1M	Full Dynamic	60.93 (-8.29)	57.67 (-7.63)	44.16 (-7.23)	54.25 (-7.72)
	Decoupled (Ours)	**69.22** (+0.00)	**63.35** (-1.95)	**53.07** (+1.68)	**61.88** (-0.09)
All attn + 2M	Full Dynamic	60.63 (-8.59)	57.44 (-7.86)	43.58 (-7.81)	53.88 (-8.09)
	Decoupled (Ours)	**69.26** (+0.04)	**63.05** (-2.25)	**52.10** (+0.71)	**61.47** (-0.50)
All layers	Full Dynamic	68.37 (-0.85)	62.98 (-2.32)	50.42 (-0.97)	60.59 (-1.38)
	Decoupled (Ours)	**68.85** (-0.37)	**64.92** (-0.38)	**52.16** (+0.77)	**61.98** (+0.01)

Table 4 | **Importance Score Ablation for Mamba Layers.** Training-free reduction at ∼25% token budget. We compare *Full Mamba Dynamic* (Eq. 1, decay) vs. *Decoupled Content Alignment* (Eq. 2, no decay). Full Mamba Dynamic degrades performance by up to 8.09 points when Mamba layers are used, as cumulative decay $\prod_u \bar{A}_u$ suppresses early visual tokens regardless of relevance. Removing decay preserves near-baseline accuracy and improves LVBench. (·) denotes change vs. baseline.



(i) End-to-end inference latency

(ii) Component-wise latency

Figure 3 | **Inference-latency gains from token reduction for Nemotron-Nano-V2 VL 12B. (i)** Measured on a single A100 with 256 frames; token reduction lowers end-to-end latency, achieving up to 2.71× speedup. **(ii)** Vision encoder, projector, and LLM latency for baseline vs. all-layer reduction on 256-frame input.

Importance Score Design: Decay Term Ablation. A central methodological claim of this paper is that the *Full Mamba Dynamic* (Equation (1)) is a poor importance signal compared to *Decoupled Content Alignment* (Equation (2)). Table 4 validates this claim directly. Under identical training-free reduction at ∼25%, using the full dynamic drops Avg by 7.72 (`All attn+1M`) and 8.09 (`All attn+2M`), with large LVBench degradations (−7.23, −7.81). When applied at all layers, the gap shrinks to 1.38 as decay weakens in deeper layers. In all cases, the decoupled score matches or exceeds the baseline, reflecting intrinsic scoring quality independent of finetuning. These results suggest that cumulative decay $\prod_u \bar{\boldsymbol{A}}_u$ should be excluded for token selection, leaving alignment $\bar{\boldsymbol{b}}_j^\top \boldsymbol{c}_t$ as the key signal, a principle that applies to other SSM/linear-attention hybrids.

Mamba Token-Reduction Patterns. Table 5 (Left) studies how to insert Mamba-layer reduction when applied to all attention layers (Figure 2i). Adding reduction at the second inter-attention position (`A+1M(L2)`) matches baseline (69.22), while two middle reductions (`A+2M(L2,3)`) slightly improve it (69.26, +0.04). In contrast, including the earliest position (`A+2M(L1,3)`) degrades performance (68.26, −0.96), indicating early reduction harms later processing, consistent with Section 2.

Reduction Types. Table 5 (Right) compares query-based token selection and average pooling. Query-based selection achieves the best accuracy (69.70, +0.48) with a 1.9× speedup. In contrast, replacing Mamba reduction with average pooling (`Query (T) + AvgPool (M)`) lowers VideoMME to 66.15 (−3.07), and pooling everywhere performs worst (65.67, −3.55) despite similar speedups. Overall, query-based selection better preserves task-relevant information under heavy compression.

4.5. Additional Results

Transformer Results. Table 8 (Appendix) reports train-time token reduction on Qwen3-VL 8B using the same positions as hybrids. While TTFT improves significantly (4.78 s → 1.07–1.18 s, up to 4.5×), accuracy drops more than in hybrids, especially on LVBench, with Avg decreasing by ∼3–4 points. Early-layer reduction is particularly harmful, consistent with Section 2, as stateless reduction in Transformers causes information loss. Figure 5ii (Appendix) shows similar latency trends, with up to 2.36× end-to-end speedup at 256 frames.

Comparisons with Other VLMs. Table 9 (Appendix) compares our method with proprietary models, open-source video LLMs, and prior token-selection approaches. Direct parameter-matched comparison is not always possible: no public *hybrid* video VLM exists in the 8B range, and existing 8B baselines differ in vision backbones and training. Our goal is to show that our architecture-aware reduction is effective for hybrids and competitive with strong Transformer methods. Despite these differences, our approach narrows the gap to much larger models (e.g., 72B) on LongVideoBench and LVBench while remaining competitive on VideoMME.

Reduction Pattern	Video-MME ↑
Baseline	69.22
A + 1M (L2)	69.22 (+0)
A + 2M (L2,3)	69.26 (+0.04)
A + 2M (L1,3)	68.26 (-0.96)

Reduction Type	Video-MME ↑	TTFT ↓ (s)
Baseline	69.22	4.65
Query-based only	69.70 (+0.48)	1.21 (×1.9)
Query (T) + AvgPool (M)	66.15 (-3.07)	1.18 (×1.9)
AvgPool only	65.67 (-3.55)	1.16 (×1.9)

Table 5 | **Mamba Token Reduction Patterns and Reduction Types at 25% budget**. (left) Reduction is applied to all attention (A) and selected Mamba (M) layers. (L·) denotes Mamba layer indices where reduction is applied (see Figure 2i). (right) Query-based reduction (A) is compared with average pooling (M). TTFT measures LLM-stage latency on a single A100 with 256-frame input.

Metric	No Reduction	25% Reduction	Gain
Train runtime	1251.7s	862.3s	1.45×
Wall clock	1390s	942s	1.48×
Samples/sec	0.818	1.187	1.45×
Peak GPU memory	77,436 MiB	70,664 MiB	8.7% saved
Max frames/sample	96	128	+33%

Table 6 | **Training efficiency with token reduction.** Runtime metrics are measured under the same training setup with 64-frame inputs. The maximum frames per sample are measured under the same A100 80GB memory budget.

5. Related Works

Token reduction is widely used in multimodal LLMs to reduce redundant computation, especially for long-video inputs. Prior work includes query-aware selection based on language–vision relevance [1, 2], token merging [26], and long-video compression methods such as FlexSelect, DynTok, and METok [6, 27, 5]. These approaches are largely developed for Transformer architectures and operate primarily on attention layers. Reduction strategies also vary from pruning from early layers [1, 28] to progressive schemes across layers [3, 29, 30]. Our work targets Mamba–Transformer hybrid VLMs and explicitly models their stateful information flow, using depth-dependent importance and recurrent state accumulation. A detailed review is provided in Section A.8 (Appendix).

6. Conclusion and Limitations

We study token reduction for long-video VLMs with a focus on Mamba–Transformer hybrids. We show that token importance is unstable across layers and that reduction is *stateful*, allowing information from discarded tokens to persist. We propose a progressive reduction schedule and a unified query-conditioned scoring method for attention and Mamba layers. Our approach achieves substantial speedups at aggressive budgets while maintaining near-baseline accuracy, with gains from light finetuning.

There are several limitations. The reduction schedule is fixed and not input-adaptive or learnable, and our evaluation focuses on visual token reduction in long-video benchmarks, leaving other modalities and deployment scenarios for future work. While we provide strong empirical evidence for stateful compression in hybrid models, a formal theoretical characterization remains an open direction.

7. Acknowledgments

We would like to thank the NVIDIA Nemotron team for their help on the hybrid model and the NVIDIA VILA team for their support on the multimodal LLM codebase. We also thank Orazio Gallo, Abhishek Badki, and Hang Su for insightful discussions.

References

[1] 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, 2024.

[2] 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, 2025.

[3] Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, et al. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*, 2024.

[4] Qizhe Zhang, Aosong Cheng, Ming Lu, Renrui Zhang, Zhiyong Zhuo, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20857–20867, 2025.

[5] Mengyue Wang, Shuo Chen, Kristian Kersting, Volker Tresp, and Yunpu Ma. Metok: Multi-stage event-based token compression for efficient long video understanding. *arXiv preprint arXiv:2506.02850*, 2025.

[6] Yunzhu Zhang, Yu Lu, Tianyi Wang, Fengyun Rao, Yi Yang, and Linchao Zhu. Flexselect: Flexible token selection for efficient long video understanding. *arXiv preprint arXiv:2506.00993*, 2025.

[7] NVIDIA, :, Aarti Basant, Abhijit Khairnar, Abhijit Paithankar, Abhinav Khattar, Adithya Renduchintala, Aditya Malte, Akhiad Bercovich, Akshay Hazare, Alejandra Rico, Aleksander Ficek, Alex Kondratenko, Alex Shaposhnikov, Alexander Bukharin, Ali Taghibakhshi, et al. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model, 2025.

[8] NVIDIA, :, Amala Sanjay Deshmukh, Kateryna Chumachenko, Tuomas Rintamaki, Matthieu Le, Tyler Poon, Danial Mohseni Taheri, Ilia Karmanov, Guilin Liu, Jarno Seppanen, Guo Chen, Karan Sapra, Zhiding Yu, Adi Renduchintala, Charles Wang, Peter Jin, Arushi Goel, Mike Ranzinger, Lukas Voegtle, Philipp Fischer, et al. Nvidia nemotron nano v2 vl, 2025.

[9] Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 24108–24118, 2025.

[10] Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. *Advances in Neural Information Processing Systems*, 37:28828–28857, 2024.

[11] Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Ming Ding, Xiaotao Gu, Shiyu Huang, Bin Xu, et al. Lvbench: An extreme long video understanding benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22958–22967, 2025.

[12] Guangxuan Xiao. Why stacking sliding windows can't see very far. https://guangxuanx.com/blog/stacking-swa.html, 2025.

[13] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938.

[14] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality, 2024.

[15] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[16] Shu Zhong, Mingyu Xu, Tenglong Ao, and Guang Shi. Understanding transformer from the perspective of associative memory. *arXiv preprint arXiv:2505.19488*, 2025.

[17] Samir Khaki, Junxian Guo, Jiaming Tang, Shang Yang, Yukang Chen, Konstantinos N Plataniotis, Yao Lu, Song Han, and Zhijian Liu. Sparsevila: Decoupling visual sparsity for efficient vlm inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23784–23794, 2025.

[18] Shiyu Zhao, Zhenting Wang, Felix Juefei-Xu, Xide Xia, Miao Liu, Xiaofang Wang, Mingfu Liang, Ning Zhang, Dimitris N Metaxas, and Licheng Yu. Accelerating multimodal large language models by searching optimal vision token reduction. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 29869–29879, 2025.

[19] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

[20] 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, et al. Qwen3-vl technical report, 2025.

[21] Jindong Jiang, Xiuyu Li, Zhijian Liu, Muyang Li, Guo Chen, Zhiqi Li, De-An Huang, Guilin Liu, Zhiding Yu, Kurt Keutzer, et al. Storm: Token-efficient long video understanding for multimodal llms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5830–5841, 2025.

[22] Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Video instruction tuning with synthetic data. *arXiv preprint arXiv:2410.02713*, 2024.

[23] Guo Chen, Zhiqi Li, Shihao Wang, Jindong Jiang, Yicheng Liu, Lidong Lu, De-An Huang, Wonmin Byeon, Matthieu Le, Tuomas Rintamaki, et al. Eagle 2.5: Boosting long-context post-training for frontier vision-language models. *arXiv preprint arXiv:2504.15271*, 2025.

[24] Leqi Shen, Guoqiang Gong, Tao He, Yifeng Zhang, Pengzhang Liu, Sicheng Zhao, and Guiguang Ding. Fastvid: Dynamic density pruning for fast video large language models, 2025.

[25] Qizhe Zhang, Mengzhen Liu, Lichen Li, Ming Lu, Yuan Zhang, Junwen Pan, Qi She, and Shanghang Zhang. Beyond attention or similarity: Maximizing conditional diversity for token pruning in mllms, 2025.

[26] Jeongseok Hyun, Sukjun Hwang, Su Ho Han, Taeoh Kim, Inwoong Lee, Dongyoon Wee, Joon-Young Lee, Seon Joo Kim, and Minho Shim. Multi-granular spatio-temporal token merging for training-free acceleration of video llms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23990–24000, 2025.

[27] Hongzhi Zhang, Jingyuan Zhang, Xingguang Ji, Qi Wang, and Fuzheng Zhang. Dyntok: Dynamic compression of visual tokens for efficient and effective video understanding. *arXiv preprint arXiv:2506.03990*, 2025.

[28] Jiaying Zhu, Yurui Zhu, Xin Lu, Wenrui Yan, Dong Li, Kunlin Liu, Xueyang Fu, and Zheng-Jun Zha. Visionselector: End-to-end learnable visual token compression for efficient multimodal llms. *arXiv preprint arXiv:2510.16598*, 2025.

[29] Juntao Liu, Liqiang Niu, Wenchao Chen, Jie Zhou, and Fandong Meng. Laco: Efficient layer-wise compression of visual tokens for multimodal large language models. *arXiv preprint arXiv:2507.02279*, 2025.

[30] 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 Computer Vision and Pattern Recognition Conference*, pages 19792–19802, 2025.

[31] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024.

[32] NVIDIA, :, Aaron Blakeman, Aarti Basant, Abhinav Khattar, Adithya Renduchintala, Akhiad Bercovich, Aleksander Ficek, Alexis Bjorlin, Ali Taghibakhshi, Amala Sanjay Deshmukh, Ameya Sunil Mahabaleshwarkar, Andrew Tao, Anna Shors, Ashwath Aithal, Ashwin Poojary, Ayush Dattagupta, Balaram Buddharaju, Bobby Chen, Boris Ginsburg, Boxin Wang, Brandon Norick, Brian Butterfield, Bryan Catanzaro, Carlo del Mundo, Chengyu Dong, et al. Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models, 2025.

[33] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[34] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[35] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

[36] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[37] Zhijian Liu, Ligeng Zhu, Baifeng Shi, Zhuoyang Zhang, Yuming Lou, Shang Yang, Haocheng Xi, Shiyi Cao, Yuxian Gu, Dacheng Li, et al. Nvila: Efficient frontier visual language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 4122–4134, 2025.

[38] Orr Zohar, Xiaohan Wang, Yann Dubois, Nikhil Mehta, Tong Xiao, Philippe Hansen-Estruch, Licheng Yu, Xiaofang Wang, Felix Juefei-Xu, Ning Zhang, et al. Apollo: An exploration of video understanding in large multimodal models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 18891–18901, 2025.

[39] Boqiang Zhang, Kehan Li, Zesen Cheng, Zhiqiang Hu, Yuqian Yuan, Guanzheng Chen, Sicong Leng, Yuming Jiang, Hang Zhang, Xin Li, et al. Videollama 3: Frontier multimodal foundation models for image and video understanding. *arXiv preprint arXiv:2501.13106*, 2025.

[40] Zuyan Liu, Yuhao Dong, Ziwei Liu, Winston Hu, Jiwen Lu, and Yongming Rao. Oryx mllm: On-demand spatial-temporal understanding at arbitrary resolution. *arXiv preprint arXiv:2409.12961*, 2024.

[41] Kai Hu, Feng Gao, Xiaohan Nie, Peng Zhou, Son Tran, Tal Neiman, Lingyun Wang, Mubarak Shah, Raffay Hamid, Bing Yin, et al. M-llm based video frame selection for efficient video understanding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 13702–13712, 2025.

[42] Xiaoqian Shen, Yunyang Xiong, Changsheng Zhao, Lemeng Wu, Jun Chen, Chenchen Zhu, Zechun Liu, Fanyi Xiao, Balakrishnan Varadarajan, Florian Bordes, et al. Longvu: Spatiotemporal adaptive compression for long video-language understanding. *arXiv preprint arXiv:2410.17434*, 2024.

[43] Mingze Xu, Mingfei Gao, Shiyu Li, Jiasen Lu, Zhe Gan, Zhengfeng Lai, Meng Cao, Kai Kang, Yinfei Yang, and Afshin Dehghan. Slowfast-llava-1.5: A family of token-efficient video large language models for long-form video understanding. *arXiv preprint arXiv:2503.18943*, 2025.

[44] Chuanqi Cheng, Jian Guan, Wei Wu, and Rui Yan. Scaling video-language models to 10k frames via hierarchical differential distillation. *arXiv preprint arXiv:2504.02438*, 2025.

[45] 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 Computer Vision and Pattern Recognition Conference*, pages 18992–19001, 2025.

[46] Pengfei Jiang, Hanjun Li, Linglan Zhao, Fei Chao, Ke Yan, Shouhong Ding, and Rongrong Ji. Visa: Group-wise visual token selection and aggregation via graph summarization for efficient mllms inference. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 11130–11139, 2025.

A. Appendix

A.1. The Implicit Attention Mechanism of Mamba

We derive the implicit attention structure in selective state-space models (Mamba), showing how each output token aggregates information from all preceding tokens through learned, input-dependent weights.

A.1.1. State-Space Recurrence

Consider the discrete-time state-space recurrence in Mamba-2 [14]:

$$\boldsymbol{S}_t = \bar{\boldsymbol{A}}_t \boldsymbol{S}_{t-1} + \boldsymbol{x}_t(\Delta_t \boldsymbol{b}_t)^\top, \tag{3}$$

$$\boldsymbol{y}_t = \boldsymbol{S}_t \boldsymbol{c}_t, \tag{4}$$

where $\boldsymbol{S}_t \in \mathbb{R}^{d\times n}$ is the hidden state matrix, $\bar{\boldsymbol{A}}_t = \exp(-\Delta_t \boldsymbol{A})$ is the discretized state transition (a diagonal matrix with entries in $(0,1)$), $\boldsymbol{x}_t \in \mathbb{R}^d$ is the input, $\boldsymbol{b}_t, \boldsymbol{c}_t \in \mathbb{R}^n$ are input-dependent projection vectors, and $\Delta_t > 0$ is the discretization step size.

A.1.2. Unrolling the Recurrence

Substituting Equation (3) into itself recursively:

$$\begin{aligned}\boldsymbol{S}_t &= \bar{\boldsymbol{A}}_t \boldsymbol{S}_{t-1} + \boldsymbol{x}_t(\Delta_t \boldsymbol{b}_t)^\top \\ &= \bar{\boldsymbol{A}}_t \left(\bar{\boldsymbol{A}}_{t-1}\boldsymbol{S}_{t-2} + \boldsymbol{x}_{t-1}(\Delta_{t-1}\boldsymbol{b}_{t-1})^\top\right) \\ &\quad + \boldsymbol{x}_t(\Delta_t \boldsymbol{b}_t)^\top \\ &= \sum_{j=1}^{t} \left(\prod_{u=j+1}^{t} \bar{\boldsymbol{A}}_u\right) \boldsymbol{x}_j(\Delta_j \boldsymbol{b}_j)^\top. \end{aligned} \tag{5}$$

Substituting into Equation (4):

$$\begin{aligned}\boldsymbol{y}_t = \boldsymbol{S}_t \boldsymbol{c}_t &= \sum_{j=1}^{t} \left(\prod_{u=j+1}^{t} \bar{\boldsymbol{A}}_u\right) \boldsymbol{x}_j(\Delta_j \boldsymbol{b}_j)^\top \boldsymbol{c}_t \\ &= \sum_{j=1}^{t} \underbrace{\left(\prod_{u=j+1}^{t} \bar{\boldsymbol{A}}_u\right) (\Delta_j \boldsymbol{b}_j)^\top \boldsymbol{c}_t}_{w_{t,j}} \boldsymbol{x}_j. \end{aligned} \tag{6}$$

This reveals an attention-like structure: the output $\boldsymbol{y}_t$ is a weighted combination of all past inputs $\{\boldsymbol{x}_j\}_{j=1}^t$. The scalar weights $w_{t,j}$ depend on three factors:

1. **Content alignment**: $\boldsymbol{b}_j^\top \boldsymbol{c}_t$ measures the relevance between the input at position j (encoded by $\boldsymbol{b}_j$) and the query at position t (encoded by $\boldsymbol{c}_t$).
2. **Input gating**: Δ_j controls how strongly position j writes to the state.
3. **Temporal decay**: $\prod_{u=j+1}^{t} \bar{\boldsymbol{A}}_u$ exponentially decays contributions from distant positions.

Equation (6) suggests an intuitive role for $\boldsymbol{b}$ and $\boldsymbol{c}$, we next formalize this by mapping the recurrence directly to the key-query formulation of Linear Attention. This connection justifies our use of these projections for token selection.

A.2. Token Selection via Mamba's Recurrent Updates

Following existing works in analyzing the associative memory mechanism in attention [15, 31, 16], we draw a connection between standard Softmax attention and the recurrence in state-space models. This elucidates why the projections $\boldsymbol{b}$ and $\boldsymbol{c}$ function as keys and queries, respectively, and justifies their use as an importance metric.

A.2.1. Attention via Kernel Functions

We begin by expressing softmax attention in a more general form using kernel functions [16], which will reveal its structural similarity to Mamba's recurrence. Standard softmax attention computes the output as a weighted sum of values:

$$\boldsymbol{y}_t = \sum_{j=1}^{t} \frac{\exp(\boldsymbol{q}_t^\top \boldsymbol{k}_j/\sqrt{d})}{\sum_{i=1}^{t} \exp(\boldsymbol{q}_t^\top \boldsymbol{k}_i/\sqrt{d})} \boldsymbol{v}_j, \tag{7}$$

The key observation is that the exponential term $\exp(\boldsymbol{q}_t^\top \boldsymbol{k}_j/\sqrt{d})$ can be viewed as a *kernel function* $\kappa(\boldsymbol{q}_t, \boldsymbol{k}_j)$ that measures similarity between the query and key [15]. A kernel function can always be decomposed as an inner product in some (possibly high-dimensional) feature space: $\kappa(\boldsymbol{q}, \boldsymbol{k}) = \phi(\boldsymbol{q})^\top \phi(\boldsymbol{k})$, where $\phi(\cdot)$ is a feature map. This decomposition allows us to express the attention weight as a normalized inner product in feature space. By defining the normalization factor as $Z_t = \sum_{i=1}^{t} \phi(\boldsymbol{q}_t)^\top \phi(\boldsymbol{k}_i)$, attention can be written as:

$$\boldsymbol{y}_t = \sum_{j=1}^{t} \underbrace{\frac{1}{Z_t}}_{\text{Normalization}} \underbrace{\left(\phi(\boldsymbol{q}_t)^\top \phi(\boldsymbol{k}_j)\right)}_{\text{Attention Score}} \boldsymbol{v}_j. \tag{8}$$

This reformulation reveals that attention computes a weighted sum of values, where each weight is determined by the similarity between the query and key in a feature space induced by ϕ. Different choices of the feature map ϕ yield different attention mechanisms: standard softmax attention corresponds to an exponential feature map $\phi(\boldsymbol{x}) = \exp(\boldsymbol{x}/\sqrt{d})$, while *linear attention* [15] uses the identity map $\phi(\boldsymbol{x}) = \boldsymbol{x}$, giving $\kappa(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^\top \boldsymbol{k}$. The advantage of this kernel view is

that when ϕ is simple (e.g., identity), the computation can be reorganized to avoid the quadratic complexity of standard attention.

As we show below, Mamba's recurrence can be viewed through this same lens, where the projections $\boldsymbol{b}$ and $\boldsymbol{c}$ play the roles of keys and queries, respectively. In this kernel form, the output decomposes into two factors: a normalization term (global $1/Z_t$ for attention, or temporal decay for Mamba) and a content-based alignment score $\phi(\boldsymbol{q}_t)^\top \phi(\boldsymbol{k}_j)$ that measures query-key compatibility.

A.2.2. The Implicit Attention in Mamba

Comparing this to the unrolled Mamba-2 output derived in Equation (6), we define the cumulative decay term $\alpha_{t,j}$ to represent the aggregate state transition from step j to t: $\alpha_{t,j} = \prod_{u=j+1}^{t} \bar{\boldsymbol{A}}_u$. The Mamba update rule can then be written in a form structurally identical to Equation (8):

$$\boldsymbol{y}_t = \sum_{j=1}^{t} \underbrace{\alpha_{t,j}}_{\text{Decay}} \underbrace{\left(\boldsymbol{c}_t^\top (\Delta_j \boldsymbol{b}_j)\right)}_{\text{Attention Score}} \boldsymbol{x}_j. \tag{9}$$

This reveals a direct correspondence between the components of Softmax attention and Mamba:

- **Query Mapping**: The term $\boldsymbol{c}_t$ corresponds exactly to the mapped query $\phi(\boldsymbol{q}_t)$.
- **Key Mapping**: The term $\Delta_j \boldsymbol{b}_j$ corresponds exactly to the mapped key $\phi(\boldsymbol{k}_j)$.
- **Alignment Score**: The dot product $\boldsymbol{c}_t^\top (\Delta_j \boldsymbol{b}_j)$ acts as the kernel similarity $\kappa(\boldsymbol{q}_t, \boldsymbol{k}_j)$, measuring the compatibility between position t and position j.

The primary difference lies in the weighting term: whereas Attention uses a global normalization $1/Z_t$ (ensuring weights sum to 1), Mamba uses a time-dependent decay $\alpha_{t,j}$ (ensuring older context fades away).

A.2.3. Resulting Token Selection Metric

Based on the mapping established above, we identify $\bar{\boldsymbol{b}}_j^\top \boldsymbol{c}_t$ (where $\bar{\boldsymbol{b}}_j = \Delta_j \boldsymbol{b}_j$) as a direct measure of query-visual alignment, analogous to the query-key dot product in attention. In our setting, where the input sequence consists of visual tokens followed by text (query) tokens, $\boldsymbol{c}_t$ is computed from the text token at position t, while $\bar{\boldsymbol{b}}_j$ is computed from the visual token at position j with the gating factor Δ_j absorbed.

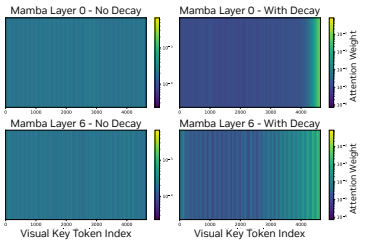


Figure 4 | Token Importance Heatmaps W/ and W/O the Decay Term on Two Mamba Layers. Left: using only the attention score $|\bar{\boldsymbol{b}}_j^\top \boldsymbol{c}_t|$ without decay (Equation (2)) produces a more distributed importance pattern across the entire sequence. Right: following Mamba's full implicit attention pattern $|w_{t,j}|$ (Equation (1)), these two layers concentrate the importance heavily on the most recent tokens. A logarithm color space is used in this figure.

Empirical Inspection of the Decay Effect. Figure 4 visualizes token importance computed with and without the decay term across Mamba layers. When following Mamba's full implicit attention pattern (Equation (9)), the cumulative decay $\alpha_{t,j} = \prod_{u=j+1}^{t} \bar{\boldsymbol{A}}_u$ causes certain layers' importance scores to concentrate heavily on tokens near the end of the visual sequence. This bias occurs when the diagonal elements of $\bar{\boldsymbol{A}}_u$ (lie in $(0, 1)$) have relatively small values, causing the weights to decay exponentially with the distance between positions j and t. Consequently, early visual tokens receive near-zero importance regardless of their actual content relevance to the query.

In contrast, when we compute importance using only the attention score component $\bar{\boldsymbol{b}}_j^\top \boldsymbol{c}_t$ without decay, the resulting heatmap shows a more distributed importance pattern across the entire sequence. Therefore, for token selection, we use the importance score:

$$s_i^{(\ell,\text{ssm})} = \frac{1}{MG} \sum_{m,g} \left| \bar{\boldsymbol{b}}_i^{(g)\top} \boldsymbol{c}_m^{(g)} \right|, \tag{10}$$

averaging over M query (text) positions and G groups.

A.3. Details of the Experimental Setup

Models. We adopt the pretrained SigLIP2 vision encoder [19] with patch size 16 at 384×384 resolution, producing 576 patch tokens. An MLP projector then pools every four adjacent patches, yielding 144 visual tokens per frame. For the LLM stage, we study two architecture families: a dense Transformer and a Mamba–Transformer hybrid. For the Transformer setting, we use the pretrained Qwen3-VL 8B [20], built on a 36-layer Qwen3 Transformer with grouped-query attention. For the hybrid set-

ting, we use Nemotron-Nano-V2 VL 12B [8], following the Nemotron-H hybrid [32] design that interleaves attention with Mamba-2 and FFN (MLP) blocks. Specifically, the 12B base architecture consists of 62 layers in total, with 28 Mamba-2, 28 FFN, and 6 self-attention layers (roughly 8%) evenly dispersed throughout the network. This setup enables us to study token reduction across both attention and Mamba layers and to compare its impact between Transformer and hybrid models. For both architectures, we reuse only the LLM backbones, integrate them with the SigLIP2 vision encoder and the random initialized vision-language projector and apply the identical multi-stage training recipe for both models to obtain the VLM models.

Training. Our training consists of four stages: (1) alignment (Stage 0), (2) SFT (Stage 1), (3) long-context finetuning (Stage 2), and (4) long-context finetuning with token-reduction (Stage 3). The maximum number of frames per sample increases from 32 (Stage 1) to 64 (Stage 2) and 128 (Stage 3) to progressively adapt the model to long video inputs; the no-reduction baseline uses 96 frames in the final stage (without reduction), which is the largest that fits in GPU memory. Following [21], Stage 0 uses 95K image–text pairs to align visual tokens to LLM by training only the MLP projector (vision encoder and LLM frozen). Stage 1 performs supervised finetuning (SFT) on a large, diverse mixture of ∼12.5M text/image/video examples (up to 32 frames). Stage 2 performs long-context finetuning on the LLaVA-Video dataset [22] (up to 64 frames). Finally, Stage 3 briefly finetunes the model with token reduction on long videos (up to 128 frames) using EAGLE-Video-110K [23], a dataset for long-video understanding that includes videos up to 3.5 hours. All stages are trained using 128 NVIDIA A100 80GB GPUs. Alignment, SFT, and long-context fine-tuning take 46 minutes, 11 days, and 25 hours, respectively.

Evaluation Benchmarks. We evaluate all token reduction techniques on three long-context video benchmarks: VideoMME [9], LongVideoBench (LongVB) [10], and LVBench [11]. VideoMME provides broad video QA; we report VideoMME in the *w/o subtitles* setting to isolate visual-temporal reasoning without relying on transcript cues. LongVB evaluates QA over long clips (up to ∼1 hour), while LVBench targets substantially longer videos (30 min–2 hrs), further emphasizing long-horizon reasoning.

A.4. Reduction Schedule Configurations

Table 7 details the reduction schedule configurations used in our experiments on the Nemotron-Nano-V2 VL 12B hybrid model [8]. The model consists of 62 layers total (30 Mamba + 30 MLP + 6 attention), where attention layers appear at positions 7, 16, 25, 34, 43, and 52 (0-indexed). In the table, α_ℓ denotes the token retention ratio at layer ℓ, i.e., the fraction of tokens kept after reduction (e.g., α=0.25 means retaining 25% of tokens). We employ two schedule types:

Sigmoid Schedule. The retention rate α_ℓ at layer ℓ follows a sigmoid function:

$$\alpha_\ell = \alpha_{\text{end}} + (\alpha_{\text{start}} - \alpha_{\text{end}}) \cdot \sigma\big(k(x_0 - \ell/L)\big), \quad (11)$$

where $\sigma(\cdot)$ is the sigmoid function, L is the total number of layers, k controls the steepness of the transition, and x_0 determines the midpoint point (as a fraction of total layers). A smaller x_0 delays reduction to later layers, implementing the low-to-high pattern that preserves more tokens early when importance scores are less reliable. Larger k produces sharper transitions between high and low retention regions. In our experiments, we fix the steepness $k = 20$ and the start value α_{start}=1.0 (no reduction initially) and the end value α_{end}=0.125 (retain 12.5% at final layers), varying $x_0 \in \{0.11, 0.24, 0.41\}$ to achieve compression rates of approximately 25%, 35%, and 50% respectively. The sigmoid schedule provides smooth, progressive transitions of the reduction rate across all layers.

Step Decay Schedule. The retention rate α_ℓ is specified explicitly for groups of layers, allowing fine-grained control over reduction patterns. For attention-only reduction, we define rates for the 6 attention positions. For example, $\alpha_{0:6}$=1.0; $\alpha_{7:61}$=0.15 means layers 0–6 retain all tokens while layers 7–61 (i.e., after the first attention layer) retain 15%. For hybrid reduction spanning multiple layer types, we typically set α_0=1.0 (no reduction at the first layer) and gradually decay across subsequent reduction layers. We hypothesize that this low-to-high sparsity levels allow the early Mamba layer to compress as much information as possible in the hidden state before the later layers starts to prune tokens. This pattern empirically yields better performance than uniform reduction variants, e.g., 1st Mamba or 2nd Mamba variants.

Layer Types. We evaluate three strategies for where to apply reduction: (1) *Attention-only*, reducing tokens only at the 6 attention layers; (2) *Mamba-only*,

Layer Type	Schedule	Reduction Layers	Comp. (%)	Parameters
Attn Only: Token reduction applied only at the 6 attention layers at positions 7, 16, 25, 34, 43, 52				
Attn Only	Step Decay	1st Attn	24.6	$\alpha_{0:6}$=1.0; $\alpha_{7:61}$=0.15
		All Attn	25.2	$\alpha_{0:6}$=1.0; $\alpha_{7:15}$=0.25, $\alpha_{16:33}$=0.20, $\alpha_{34:61}$=0.10
			34.7	$\alpha_{0:6}$=1.0; $\alpha_{7:15}$=0.50, $\alpha_{16:24}$=0.40, $\alpha_{25:33}$=0.30, $\alpha_{34:42}$=0.20, $\alpha_{43:61}$=0.10
			50.1	$\alpha_{0:6}$=1.0; $\alpha_{7:15}$=0.65, $\alpha_{16:24}$=0.60, $\alpha_{25:33}$=0.50, $\alpha_{34:42}$=0.40, $\alpha_{43:51}$=0.30, $\alpha_{52:61}$=0.20
Mamba Only: Token reduction only applied once at one early Mamba layer				
Mamba Only	Step Decay	1st Mamba	25.0	$\alpha_{0:61}$=0.25
		2nd Mamba	25.5	α_0=1.0; $\alpha_{1:61}$=0.23
Mamba+Attn (Step Decay): Interleaving reduction with both attention layers and Mamba layers				
Mamba+Attn	Step Decay	All Attn+1M	25.4	α_0=1.0; $\alpha_{1:33}$: 0.32→0.15 (6 attn + 1 mamba between each attn pair)
		All Attn+2M	25.4	α_0=1.0; $\alpha_{1:33}$: 0.32→0.15 (6 attn + 2 mamba between each attn pair)
Mamba+Attn (Sigmoid): All-layer reduction with sigmoid schedule $\alpha_\ell = \alpha_{\text{end}} + (\alpha_{\text{start}} - \alpha_{\text{end}}) \cdot \sigma(k(x_0 - \ell/L))$.				
Mamba+Attn	Sigmoid	All	25.1	k=20, x_0=0.11; α_{start}=1.0, α_{end}=0.125
			35.0	k=20, x_0=0.24; α_{start}=1.0, α_{end}=0.125
			50.2	k=20, x_0=0.41; α_{start}=1.0, α_{end}=0.125

Table 7 | Complete reduction schedule configurations for hybrid models (Nemotron-Nano-V2 VL 12B). Following the official model configuration [8], we include layer indices for MLP layers when explaining the reduction parameters. **Layer Type** indicates the layer types performing token reduction: "Attn Only" applies reduction only at the 6 attention layers; "Mamba Only" includes reduction starting from a single Mamba layer; "Mamba+Attn" applies reduction at a subset or all 62 layers (excluding MLP layers). **Reduction Layers** specifies the layer subset: "1st Attn" reduces tokens starting from the first attention layer; "All attn" applies different rates at each of the 6 attention layers; "1st Mamba" and "2nd Mamba" for single Mamba layer configurations; "All Attn+1M" and "All Attn+2M" for 6 attention layers plus 1 or 2 Mamba reduction layers between each attention pair; "All" for all layers with progressive reduction. Layer indices are 0-indexed. All configurations enforce a minimum of 144 tokens to prevent reduction on single image data.

reducing at early Mamba layers but maintain a relatively low sparsity on deeper layers; and (3) *Hybrid (Mamba+Attn)*, applying reduction at both attention and selected Mamba layers. For hybrid schedules, the notation "All Attn+1M" indicates 1 Mamba reduction between each pair of attentions, so there are 6 attention layers plus 7 Mamba layers. Similarly, "All Attn+2M" indicates 2 Mamba layers between each pair of attentions. All configurations enforce a minimum of 144 tokens to preserve full tokens for image data in training.

A.5. End-to-End Latency Across Frame Counts

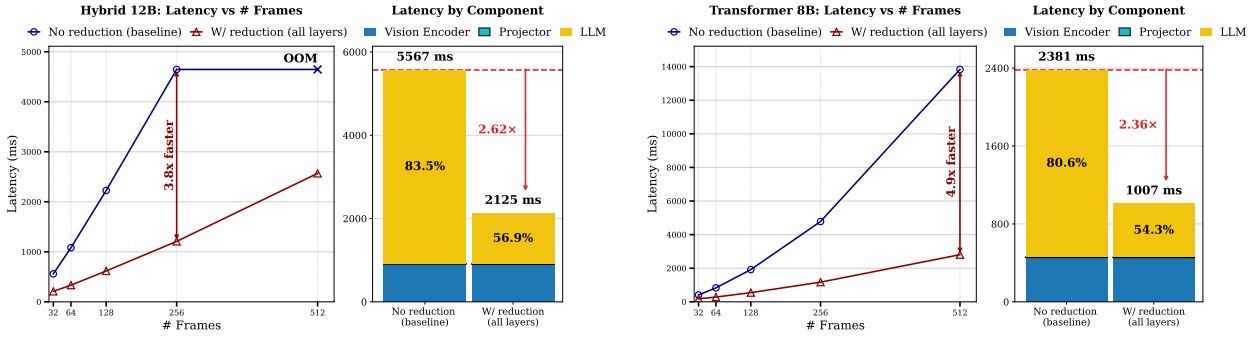
Table 10 reports end-to-end inference latency (vision encoder + projector + LLM, in seconds) for Nemotron-Nano-V2 VL 12B on a single NVIDIA A100, sweeping the number of input frames from 16 to 256. We compare three configurations: no reduction, attention-only token reduction at ∼25% token budget, and the full attention+Mamba (`All`-layer) reduction at ∼25%. The all-layer variant achieves a 1.6× speedup at 16 frames, growing to 2.4–2.6× for 64–256 frames. The relative gain widens with input length because the LLM prefilling cost grows superlinearly while the vision encoder cost scales linearly with the number of frames; for long videos the LLM stage dominates total runtime, so its acceleration translates almost directly into the end-to-end metric. This complements Figure 3i (LLM-stage TTFT only) by accounting for the full prefill pipeline.

A.6. Training-Time Efficiency

Token reduction is not only an inference-time optimization. We also benchmark training throughput and memory under `All`-layer reduction at ∼25%

Token Reduction	Comp. Rate (%)	VideoMME (1∼60m)	LongVB (8s∼60m)	LVBench (30m∼2h)	Avg	TTFT (s)
Baseline	100	69.89	66.64	50.94	62.49	4.78
Train-Time Reduction						
1st layer	25.0	59.52 (-10.37)	55.95 (-10.69)	38.28 (-12.66)	51.25 (-11.24)	1.07 (×4.5)
6 layers	25.1	68.41 (-1.48)	**64.62 (-2.02)**	43.19 (-7.75)	58.74 (-3.75)	1.18 (×4.1)
13 layers	25.1	67.93 (-1.96)	63.65 (-2.99)	44.29 (-6.65)	58.62 (-3.87)	1.09 (×4.4)
20 layers	25.0	68.00 (-1.89)	63.87 (-2.77)	44.16 (-6.78)	58.68 (-3.81)	1.09 (×4.4)
All	25.5	**68.52 (-1.37)**	64.32 (-2.32)	**45.45 (-5.49)**	**59.43 (-3.06)**	1.17 (×4.1)

Table 8 | **Token Reduction with Different Reduction Patterns for Qwen3-VL 8B.** Train-time reduction is applied at selected LLM layers (1st, 6/13/20, or all), using the same relative layer positions as the hybrid setup. "Comp. Rate" is remaining tokens; (·) shows change from baseline. TTFT is LLM-stage latency on a single A100 with a 256-frame input.



(i) **Nemotron-Nano-V2 VL 12B (hybrid).** (ii) **Qwen3-VL 8B (Transformer).**

Figure 5 | **Latency Analysis. (left)** LLM-stage latency vs. frames on a single A100, with/without token reduction. **(right)** Component-wise latency (vision encoder / projector / LLM) on a 128-frame VideoMME input.

budget against no-reduction training, with 64-frame video samples and otherwise identical training resources. The hybrid model's training run time drops from 1251.7 s to 862.3 s (1.45× speedup), wall-clock time drops from 1390 s to 942 s (1.48×), and throughput increases from 0.818 to 1.187 samples/sec. Peak GPU memory drops from 77,436 MiB to 70,664 MiB, a saving of ∼6.6 GB (8.7%). On A100 80 GB, the no-reduction baseline supports at most 96 frames per sample, while the reduced variant supports 128 frames in the same memory budget, enabling 33% more visual context per sample. Token reduction is therefore an essential lever for scaling long-video training, not merely inference.

A.7. Impact Statements

This paper presents work whose goal is to advance the field of machine learning by improving the efficiency of long-video vision–language models. The primary intended impact is reduced inference latency and compute/energy cost for long-context video understanding. Beyond this, we do not anticipate additional societal consequences that require specific discussion.

A.8. Related Works

Token reduction is widely used in multimodal LLMs to reduce redundant computation, especially for long video inputs. Token pruning methods are often grouped into query-agnostic approaches (based on visual similarity) and query-aware approaches (conditioned on language). Query-aware methods typically score tokens using text-to-vision attention or cross-modal similarity. FastV [1] prunes tokens using early attention signals, and SparseVLM [2] prunes visually irrelevant tokens using cross-attention scores. Instead of pruning, token merging aggregates redundant tokens, such as multi-granular spatio-temporal token merging [26]. Most of these methods are designed around attention layers and do not explicitly model token importance inside non-attention blocks, which is important for hybrid architectures.

Many works focus on long-video understanding by reducing temporal redundancy. FlexSelect [6], DynTok [27], and DyCoke [45] select or compress tokens for long videos, and METok [5] proposes multi-stage event-based compression. Our work also targets long-context video, but focuses on hybrid models and uses

Model Type	Model / Method	Size	VideoMME (1∼60m)	LongVB (8s∼60m)	LVBench (30m∼2h)
Proprietary Models					
Transformer	GPT-4o [33]	–	71.9	66.7	34.7
	Gemini-1.5-Pro [34]	–	75.0	64.0	33.1
Uniform Token Selection					
Transformer	LLaVA-Onevision [35]	7B	58.2	56.4	–
	Qwen2-VL [36]	7B	63.3	55.6	42.4
	NVILA [37]	8B	64.2	57.7	–
	Apollo [38]	7B	61.3	58.5	–
	VideoLLaMA3 [39]	7B	66.2	59.8	45.3
	Oryx-1.5 [40]	34B	67.3	62.0	30.8
Adaptive Token Selection/Reduction					
Transformer	MLLM-FS [41]	7B	58.7	57.0	–
	LongVU [42]	7B	60.6	–	–
	SF-LLaVA-1.5 [43]	7B	63.9	62.5	45.3
	ViLAMP [44]	7B	67.5	61.2	45.2
	Qwen2.5 VL+SparseVILA [17]	7B	66.3	60.1	–
	Qwen2.5 VL+FlexSelect [6]	7B	68.2	62.4	51.2
	LLaVA-Video+FlexSelect [6]	7B	68.9	61.9	52.9
	Qwen2.5+FlexSelect [6]	72B	**74.4**	**66.4**	**56.6**
Our Token Reduction					
Transformer	Qwen3 VL + All layer reduction	8B	68.52	64.32	45.45
Hybrid	Nemotron-Nano-V2 VL + All layer reduction	12B	**69.70**	**66.04**	**54.29**

Table 9 | **Comparison on Long-Video Benchmarks.** We report scores on VideoMME (w/o subtitles), LongVB (LongVideoBench; 8s∼60m), and LVBench (30m∼2h) for proprietary, open-source LLMs with uniform/adaptive token selection, and our method. Ours use all-layer reduction for Qwen3-VL (Transformer) and Nemotron-Nano-V2 VL (Hybrid).

Frames	16	32	64	128	256
No Reduction (100%)	0.349	0.677	1.313	2.714	5.567
Attention-only (25%)	0.179	0.287	0.520	1.033	2.052
Attn+Mamba `All` (25%)	0.214	0.327	0.564	1.080	2.125
Speedup (Attn+Mamba vs. baseline)	1.6×	2.1×	2.3×	2.5×	2.6×

Table 10 | **End-to-End Inference Latency** for Nemotron-Nano-V2 VL 12B on a single NVIDIA A100, in seconds (vision encoder + projector + LLM). Token reduction at a ∼25% budget yields a 1.6× speedup at 16 frames and 2.4–2.6× speedup at 64–256 frames; the gap widens with frame count because LLM prefilling cost grows super-linearly while vision encoder cost is linear.

depth-dependent token-importance behavior to design reduction schedules.

Another distinction is where reduction is applied. Some methods prune once early (before the LLM or at early layers) and keep the reduced set fixed. For example, FastV [1] prunes after early layers, and VisionSelector [28] selects tokens before the LLM. FlexSelect [6] and VISA [46] also produce a reduced token set for inference. Other methods prune progressively across layers to reduce the sensitivity of early pruning. PyramidDrop [3] drops tokens across depth, and LaCo [29], VisionZip [30], DynTok [27], and METok [5] apply multi-layer or multi-stage reduction. Our method follows this direction and uses a progressive schedule that keeps more tokens early and prunes more later.