SFPruner: Single-Forward Visual Token Subset Selection for Resource-Efficient Multimodal Foundation Model Inference

Jouwon Song [1] **Woohyeong Kim** [2] **Seungjae Baek** [1] **Kyeongbo Kong** [2]

Abstract

High-resolution multimodal foundation models spend a large fraction of inference compute on visual tokens, making visual-token pruning a practical mechanism for resource-efficient deployment. Existing pruning methods expose a recurring trade-off: fast heuristic methods add little selection overhead but weakly control redundancy, while diversity-aware subset-optimization methods better preserve non-redundant evidence but often rely on sequential greedy search, which can erase part of the wall-clock benefit of pruning. We propose SFPruner, a single-pass approximation to redundancy-aware visual-token subset selection. Instead of constructing the subset iteratively, SFPruner builds redundancy control into the scoring stage through semantics-guided ridge leverage and ranking-based directional masking, enabling one-shot Top-K selection under a target token budget. Across image and video MLLMs, SFPruner preserves competitive quality–resource trade-offs while substantially reducing selection overhead; for example, compared with CDPruner on Qwen2.5-VL at a 512-token budget, selection latency decreases from 112.4 ms to 2.5 ms while retaining 99.0% relative performance. These results show that visual-token pruning must optimize not only which tokens are retained, but also how efficiently the retained subset is selected.

1. Introduction

Recent multimodal large language models (MLLMs) (Liu et al., 2023; Wang et al., 2024; Liu et al., 2024a) perform fine-grained reasoning over high-resolution images and long videos, but dynamic-resolution models such as LLaVA-NeXT (Liu et al., 2024b) and Qwen-VL (Wang et al., 2024) often produce thousands of visual tokens per input. Because self-attention scales quadratically with sequence length, this visual-token growth creates substantial latency and memory bottlenecks during inference. Visual tokens are therefore a natural unit for resource allocation: reducing the retained visual sequence can lower attention cost, memory pressure, and prefill latency. The key challenge is to select a compact subset that preserves task-relevant, non-redundant visual evidence while keeping the selection procedure lightweight enough to preserve end-to-end speedups.

Vision token pruning addresses this budgeted subset-selection problem by retaining informative tokens before expensive attention operations (Xing et al., 2024; Chen et al., 2024; Zhang et al., 2025b), yet existing methods often trade selection efficiency for redundancy control (Fig. 1). *Heuristic* methods rank tokens using attention or similarity signals (Zhang et al., 2024a; Shang et al., 2025; Yang et al., 2025; Wen et al., 2025; Zhang et al., 2024b); they are fast, but weakly control spatial redundancy and can degrade under aggressive pruning. Merging variants such as VisionZip (Yang et al., 2025) partially recover accuracy by averaging discarded tokens into retained neighbors, but this can homogenize local features and blur fine-grained cues (Wen et al., 2025). *Subset-optimization* methods (Zhang et al., 2025a; Alvar et al., 2025; Zhang et al., 2026) preserve diversity through combinatorial objectives such as DPP, MMDP, or maximal independent set. Many of these objectives are optimized with sequential greedy approximations, whose K-step updates make selection latency grow with the retention budget. As a result, theoretical FLOP reductions do not always translate into measured wall-clock gains in high-resolution regimes (Fig. 1, right). This motivates a different design goal: token pruning should optimize both the quality of the retained subset and the cost of finding that subset.

We propose **Single-Forward Pruner (SFPruner)**, a single-forward approximation to redundancy-aware visual-token subset selection. Rather than constructing the subset iteratively, SFPruner embeds redundancy control directly into the scoring space. SFPruner adapts the *ridge leverage score* (RLS) (Musco & Musco, 2017; McCurdy, 2018) to estimate feature-space uniqueness with a ridge solve, avoiding per-

[1]LG Electronics, Seoul, Republic of Korea [2]Pusan National University, Busan, Republic of Korea. Correspondence to: Kyeongbo Kong <kbkong@pusan.ac.kr>.

International Conference on Machine Learning, AdaptFM: Resource-Adaptive Foundation Model Inference, Seoul, South Korea.

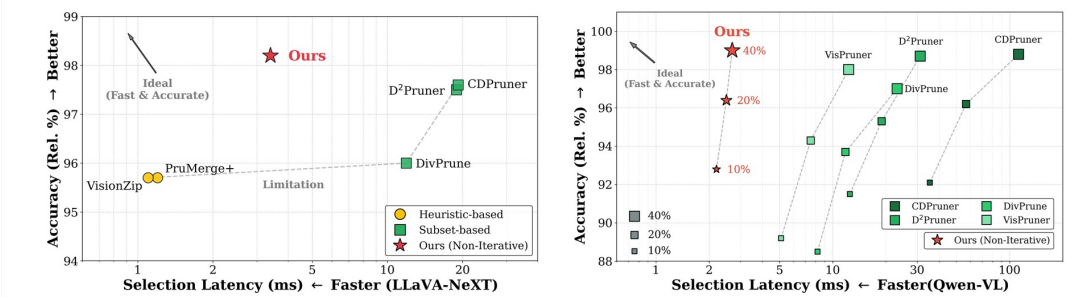

Figure 1. **Left:** LLaVA-NeXT—Rel. (%) vs. selection latency (ms, log scale): fast heuristics vs. slower subset optimizers (dashed limitation); SFPruner (red) moves toward the ideal fast-and-accurate corner. **Right:** Qwen2.5-VL at three retention ratios (marker size)—subset methods trade large latency for Rel.; SFPruner stays at few-ms latency.

step subset updates. In one selection pass, it combines two complementary mechanisms. First, *covariance-level redundancy attenuation* via semantics-guided ridge leverage fuses instruction relevance and visual saliency to downweight dominant covariance directions before selection. Second, *ranking-based directional masking* resolves residual pairwise overlap through asymmetric similarity competition: higher-scoring tokens suppress redundant lower-scoring alternatives using parallel tensor operations, removing the need for sequential state updates. On Qwen2.5-VL (Bai et al., 2025), compared with CDPruner at a 512-token budget, SFPruner reduces selection latency from 112.4 ms to 2.5 ms while retaining up to 99.0% relative performance. These results suggest that practical visual-token pruning requires redundancy-aware subset selection whose own overhead is small relative to the inference savings it enables.

2. Proposed Method: Single-Forward Pruner (SFPruner)

SFPruner performs one-pass scoring via semantic guidance (Section 2.1), semantics-guided ridge leverage (Section 2.2), and directional masking (Section 2.3); Figure 2 summarizes the pipeline.

2.1. Semantic Guidance: Textual Relevance and Visual Saliency

We first compute a semantic guidance score per token from instruction relevance and visual saliency. Let $V \in \mathbb{R}^{N\times D}$ stack L_2-normalized visual token rows v_i, and $q \in \mathbb{R}^{1\times D}$ be the projected text embedding.

Textual relevance. We measure instruction relevance using cosine similarity between each visual token and the text embedding, followed by temperature-scaled normalization:

$$S_{\text{rel},i} = \frac{\exp\big(\tau\, v_i q^\top/(\|v_i\|\|q\|)\big)}{\sum_{j=1}^{N}\exp\big(\tau\, v_j q^\top/(\|v_j\|\|q\|)\big)}, \qquad (1)$$

where τ controls sharpness.

Visual saliency. To capture instruction-agnostic visual importance, we use a saliency score from vision-encoder self-attention, i.e., attention from a global token (e.g., CLS) to each visual token. When no canonical CLS token exists (e.g., Qwen2.5-VL (Bai et al., 2025)), we use a proxy obtained by aggregating self-attention weights across spatial tokens:

$$S_{\text{attn},i} = \text{Attention}(\text{CLS} \to v_i). \qquad (2)$$

Fusion and normalization. We linearly fuse relevance and saliency,

$$S_{\text{guide},i} = \alpha S_{\text{rel},i} + (1-\alpha)S_{\text{attn},i}, \qquad \alpha \in [0,1], \quad (3)$$

using only S_{attn} when no text encoder is available. For stable scaling across inputs we min–max normalize:

$$\tilde{S}_{\text{guide},i} = \frac{S_{\text{guide},i} - \min_j S_{\text{guide},j}}{\max_j S_{\text{guide},j} - \min_j S_{\text{guide},j} + \epsilon}. \qquad (4)$$

The map $\tilde{S}_{\text{guide}}$ feeds SG-RLS in Section 2.2.

2.2. Covariance-Level Attenuation: Semantics-Guided Ridge Leverage Score

Instead of sequential residual updates, we attenuate redundancy globally with ridge-regularized leverage. We define the feature covariance matrix as

$$C = V^\top V \in \mathbb{R}^{D\times D}, \qquad (5)$$

and the ridge leverage score of token i as

$$\ell_i = v_i (C + \lambda I_D)^{-1} v_i^\top, \qquad (6)$$

where $\lambda > 0$ is a small regularization constant. The inverse covariance attenuates dominant eigendirections, assigning lower leverage to globally redundant tokens and higher scores to structurally distinctive ones. To incorporate task-awareness, we modulate this structural score with the semantic guidance from Section 2.1:

$$S_{\text{SG-RLS},i} = \ell_i \cdot \tilde{S}_{\text{guide},i}. \qquad (7)$$

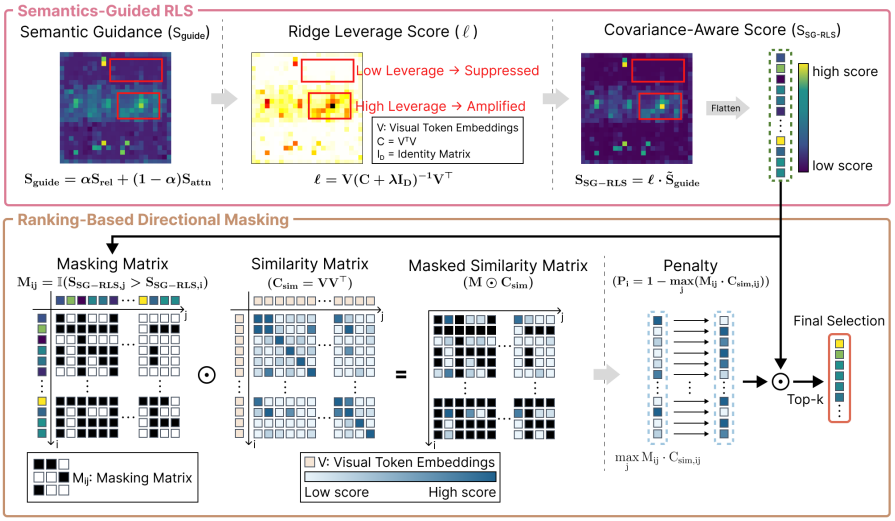

Figure 2. **Pipeline.** SFPruner computes semantics-guided ridge leverage scores, applies ranking-based directional masking to suppress tokens covered by higher-scoring neighbors, and performs one-shot Top-K selection.

Thus, ridge leverage captures global structural uniqueness, while semantic guidance suppresses task-irrelevant outliers.

The computation uses parallel matrix operations and a single ridge solve, avoiding high-resolution inversions and eliminating iterative subset updates.

2.3. Ranking-Based Directional Masking

Beyond covariance-level control in SG-RLS, we add pairwise exclusion, where high-scoring tokens suppress near-duplicate alternatives based on $C_{\text{sim}} = VV^\top \in \mathbb{R}^{N\times N}$.

We define a directional masking mechanism based on the SG-RLS scores: for each token i, suppression is applied only from tokens with strictly higher scores. Formally,

$$M_{ij} = \mathbb{I}\big(S_{\text{SG-RLS},j} > S_{\text{SG-RLS},i}\big), \tag{8}$$

with $M_{ii} = 0$. The directional suppression penalty is

$$P_i = 1 - \max_j\big(M_{ij} \cdot C_{\text{sim},ij}\big), \tag{9}$$

so $P_i \le 1$ (the max includes 0 from M_{ii}). The final score is

$$S_{\text{final},i} = S_{\text{SG-RLS},i} \cdot P_i, \tag{10}$$

and we select the top K tokens under a deployment-specified token budget. All steps are implemented with parallel GEMMs, element-wise comparisons, and row reductions.

3. Experiments

Setup. We evaluate SFPruner on LLaVA-NeXT-7B (Liu et al., 2024b), Qwen2.5-VL-7B (Bai et al., 2025), and LLaVA-Video-7B (Zhang et al., 2024c), covering multi-patch, dense single-sequence, and multi-frame visual-token regimes. We report task accuracy and relative performance retention (Rel. %) against the corresponding unpruned backbone. Unless otherwise stated, SFPruner uses α=0.5, temperature T=20, and ridge regularization λ=0.005; for Qwen2.5-VL, which lacks a canonical CLS token, we use the visual-saliency proxy only. Full experimental protocols, additional image and video results, extreme-scale profiling, and robustness analyses are provided in Appendices A–H.

3.1. Quality–Latency Trade-off

We evaluate Qwen2.5-VL-7B, a dense single-sequence setting where all high-resolution visual tokens compete within one attention window. This regime is sensitive to pruning overhead, since selection cost can offset the inference savings from token reduction.

As shown in Table 1, subset-optimization methods retain strong quality but incur high selection latency. At 40% retention, CDPruner, DivPrune, and D^{2}Pruner require 112.4 ms, 23.1 ms, and 31.2 ms, respectively, with lower latency at 20% retention due to their K-dependent greedy selection. In contrast, SFPruner remains at 2.5 ms for both budgets by using parallel covariance-level attenuation and ranking-based directional masking without iterative subset updates.

SFPruner also achieves the best aggregate Rel. at both budgets: 99.0% at 40% retention and 96.5% at 20% retention. Thus, SFPruner preserves redundancy-aware selection quality while keeping selection latency close to lightweight heuristics, supporting practical visual-token pruning for deployment.

3.2. End-to-End Scalability

We next test whether selection efficiency translates into realized wall-clock gains. Table 2 profiles Qwen2.5-VL under

Table 1. **Qwen2.5-VL-7B results.** Rel. (%) is relative to the unpruned model; **sel.** is selection time (ms). SFPruner achieves the best Rel. with 2.5 ms selection at both budgets.

Method	TextVQA	AI2D	HBench	MME	MMB	MMB^{CN}	POPE	MMStar	Rel.	sel.
Vanilla (100% tokens)										
Qwen2.5-VL-7B	85.3	80.8	64.3	2316.0	79.7	81.8	86.4	56.7	100.0	–
Retain 40% (∼512 tokens avg.)										
PruMerge (Shang et al., 2025)	82.1	79.8	62.6	2287.0	78.4	80.0	86.0	55.6	98.1	1.2
VisPruner (Zhang et al., 2024a)	82.6	79.9	61.7	2297.5	78.7	79.9	85.4	55.3	98.0	12.3
VisionZip (Yang et al., 2025)	82.6	79.5	61.2	2306.5	78.3	80.5	86.1	55.2	98.0	1.1
DivPrune (Alvar et al., 2025)	82.3	78.6	61.4	2230.6	78.5	80.0	84.5	54.7	97.0	23.1
CDPruner (Zhang et al., 2025a)	83.7	79.6	63.4	2302.2	79.2	80.3	86.1	55.9	98.8	112.4
D^2Pruner (Zhang et al., 2026)	83.2	82.9	63.7	2305.3	78.2	79.6	85.3	54.9	98.7	31.2
SFPruner	**84.0**	79.9	63.1	**2319.3**	79.0	79.7	**86.5**	**55.9**	**99.0**	**2.5**
Retain 20% (∼256 tokens avg.)										
PruMerge (Shang et al., 2025)	74.7	77.2	59.1	2280.1	77.6	77.7	84.5	52.8	94.6	1.1
VisPruner (Zhang et al., 2024a)	75.3	77.4	59.0	2213.9	77.5	78.3	84.5	52.2	94.3	7.5
VisionZip (Yang et al., 2025)	75.9	77.6	59.1	2276.5	77.4	78.5	84.5	53.5	95.1	1.1
DivPrune (Alvar et al., 2025)	76.5	76.6	58.1	2203.7	76.9	77.5	83.6	51.8	93.7	11.8
CDPruner (Zhang et al., 2025a)	79.1	78.4	61.1	2264.5	77.9	79.5	84.9	53.4	96.2	56.7
D^2Pruner (Zhang et al., 2026)	78.6	79.4	60.2	2280.5	76.1	78.6	83.0	52.1	95.3	18.9
SFPruner	**79.8**	78.3	60.2	**2295.7**	**78.1**	79.4	**85.7**	**53.7**	**96.5**	**2.5**

Table 2. **Extreme-scale hardware profiling on Qwen2.5-VL** with N=9,216 visual tokens on full MME.

Method	FLOPs (T) ↓	Memory (MB) ↓	Pruning (ms) ↓	Prefill (ms) ↓	Infer (s) ↓
Full Seq.	148	19,749	–	1813	6101
DivPrune (Alvar et al., 2025)	44.4	18,512	458	1547	5387
CDPruner (Zhang et al., 2025a)	44.4	18,736	576	1658	5787
SFPruner	44.4	**18,432**	**28**	**1114**	**3601**

an extreme high-resolution setting with N=9,216 visual tokens and a 30% retention budget. Although DivPrune and CDPruner reduce theoretical FLOPs to the same level as SFPruner, their selection overhead remains large: 458 ms and 576 ms, respectively. Consequently, their total inference times improve only modestly over the full sequence.

SFPruner completes pruning in 28 ms, reduces prefill latency from 1813 ms to 1114 ms, and cuts total inference time from 6101 s to 3601 s. This corresponds to a 1.69× end-to-end speedup, compared with 1.13× for DivPrune and 1.05× for CDPruner. The result shows that SFPruner's single-forward design converts token reduction into actual deployment speedup rather than merely reporting lower token counts.

3.3. Ablation Study

Finally, we isolate the two components of SFPruner on LLaVA-NeXT-7B with 320 retained tokens per image. As shown in Table 3, naive Top-K selection is fastest but leaves local redundancy unresolved, reaching only 96.9% Rel. Sequential selection improves Rel. to 98.0%, but its per-step state updates increase selection time to 18.6 ms. Directional masking recovers nearly the same retention, 97.9%, in only 0.8 ms by suppressing lower-scoring neighbors in parallel.

Adding covariance-level attenuation through SG-RLS further improves Rel. to 98.3%, with consistent gains on GQA and POPE. The ridge computation increases total selection latency to 2.3 ms, still far below sequential search. Together, these results show that SFPruner's two design choices are complementary: SG-RLS improves score quality, while directional masking approximates iterative redundancy exclusion without introducing a K-dependent loop.

Table 3. **Ablation on LLaVA-NeXT-7B** with 320 retained tokens per image. SG uses $\tilde{S}_{\text{guide}}$ only; SG-RLS multiplies $\tilde{S}_{\text{guide}}$ by ridge leverage ℓ_i.

Scoring	Selection	GQA	TextVQA	POPE	Rel. (%)	Time (ms) ↓
SG ($\tilde{S}_{\text{guide}}$)	Top-K	60.3	57.8	85.5	96.9	**0.7**
SG ($\tilde{S}_{\text{guide}}$)	Sequential	61.2	58.2	86.3	98.0	18.6
SG ($\tilde{S}_{\text{guide}}$)	Dir. mask	61.2	**58.3**	86.4	97.9	0.8
SG-RLS	**Dir. mask**	**61.5**	**58.3**	**86.7**	**98.3**	2.3

4. Conclusion

We revisited visual token pruning as a budgeted subset-selection problem for resource-efficient high-resolution MLLM inference. While subset-optimization methods provide strong redundancy control, their iterative greedy loops introduce selection overhead that grows with the target budget and can erode the wall-clock benefits of token reduction. We proposed **SFPruner**, a non-iterative framework that embeds redundancy-aware subset selection into structured tensor operations. **SG-RLS** attenuates covariance-level redundancy by rebalancing dominant directions under semantic guidance, while **ranking-based directional masking** suppresses residual pairwise overlap without sequential subset construction. Together, these components provide redundancy-aware token selection with low selection overhead, showing that practical token-level adaptation requires both a high-quality retained subset and a lightweight selection policy.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2024-00456152). This work was also supported by LG Electronics. The authors gratefully acknowledge the Cluster Server for Computational Science at Pusan National University for providing computational resources.

References

Alvar, S. R., Singh, G., Akbari, M., and Zhang, Y. DivPrune: Diversity-based visual token pruning for large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9392–9401, 2025.

Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Chen, L., Zhao, H., Liu, T., Bai, S., Lin, J., Zhou, C., and Chang, B. 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*, pp. 19–35, 2024.

Fu, C., Dai, Y., Luo, Y., Li, L., Ren, S., Zhang, R., Wang, Z., Zhou, C., Shen, Y., Zhang, M., et al. Video-MME: The first-ever comprehensive evaluation benchmark of multi-modal LLMs in video analysis. In *CVPR*, 2025.

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6904–6913, 2017.

Hudson, D. A. and Manning, C. D. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6700–6709, 2019.

Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Liu, H., Li, C., Wu, Q., and Lee, Y. J. Visual instruction tuning. *Advances in Neural Information Processing Systems*, 36:34892–34916, 2023.

Liu, H., Li, C., Li, Y., Li, B., Zhang, Y., Shen, S., and Lee, Y. J. LLaVA-NeXT: Improved reasoning, OCR, and world knowledge, 2024b.

Lu, P., Mishra, S., Xia, T., Qiu, L., Chang, K.-W., Zhu, S.-C., Tafjord, O., Clark, P., and Kalyan, A. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022.

McCurdy, S. Ridge regression and provable deterministic ridge leverage score sampling. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Musco, C. and Musco, C. Recursive sampling for the Nyström method. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Shang, Y., Cai, M., Xu, B., Lee, Y. J., and Yan, Y. LLaVA-PruMerge: Adaptive token reduction for efficient large multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Singh, A., Natarajan, V., Shah, M., Jiang, Y., Chen, X., Batra, D., Parikh, D., and Rohrbach, M. Towards VQA models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8317–8326, 2019.

Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., Bai, J., Chen, K., Liu, X., Wang, J., Ge, W., et al. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

Wen, Z., Gao, Y., Wang, S., Zhang, J., Zhang, Q., Li, W., He, C., and Zhang, L. Stop looking for important tokens in multimodal language models: Duplication matters more. *arXiv preprint arXiv:2502.11494*, 2025.

Wu, H., Li, D., Chen, B., and Li, J. LongVideoBench: A benchmark for long-context interleaved video-language understanding, 2024.

Xing, L., Huang, Q., Dong, X., Lu, J., Zhang, P., Zang, Y., Cao, Y., He, C., Wang, J., Wu, F., et al. PyramidDrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*, 2024.

Yang, S., Chen, Y., Tian, Z., Wang, C., Li, J., Yu, B., and Jia, J. VisionZip: Longer is better but not necessary in vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19792–19802, 2025.

Yin, S., Fu, C., Zhao, S., Li, K., Sun, X., Xu, T., and Chen, E. A survey on multimodal large language models. *National Science Review*, 11(12):nwae403, 2024.

Zhang, E., Yu, F., Wu, A., Wen, Z., Yan, K., Ding, S., Qi, B., and Zhang, L. D^2Pruner: Debiased importance and structural diversity for MLLM token pruning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

Zhang, Q., Cheng, A., Lu, M., Zhang, R., Zhuo, Z., Cao, J., Guo, S., She, Q., and Zhang, S. Beyond text-visual attention: Exploiting visual cues for effective token pruning in VLMs. *arXiv preprint arXiv:2412.01818*, 2024a.

Zhang, Q., Cheng, A., Lu, M., Zhuo, Z., Wang, M., Cao, J., Guo, S., She, Q., and Zhang, S. [CLS] attention is all you need for training-free visual token pruning: Make VLM inference faster. *arXiv e-prints*, 2024b.

Zhang, Q., Liu, M., Li, L., Lu, M., Zhang, Y., Pan, J., She, Q., and Zhang, S. Beyond attention or similarity: Maximizing conditional diversity for token pruning in MLLMs. *arXiv preprint arXiv:2506.10967*, 2025a.

Zhang, Y., Wu, J., Li, W., Li, B., Ma, Z., Liu, Z., and Li, C. Video instruction tuning with synthetic data, 2024c.

Zhang, Y., Fan, C.-K., Ma, J., Zheng, W., Huang, T., Cheng, K., Gudovskiy, D., Okuno, T., Nakata, Y., Keutzer, K., et al. SparseVLM: Visual token sparsification for efficient vision-language model inference. In *International Conference on Machine Learning*, 2025b.

Zhou, J., Shu, Y., Zhao, B., Wu, B., Xiao, S., Yang, X., Xiong, Y., Zhang, B., Huang, T., and Liu, Z. MLVU: A comprehensive benchmark for multi-task long video understanding. *arXiv preprint arXiv:2406.04264*, 2024.

Appendix Overview

- **Appendix A: Experimental Setup.** Models, benchmarks, baselines, evaluation metrics, implementation details, and token-budget conventions.
- **Appendix B: Full Image-Benchmark Results.** Full LLaVA-NeXT-7B and Qwen2.5-VL-7B results, together with the core component ablation.
- **Appendix C: Full Video-Benchmark Results.** LLaVA-Video-7B results across Video-MME, MLVU, LongVideoBench, and MMVBench.
- **Appendix D: Extreme Scalability Stress Test.** High-resolution Qwen2.5-VL profiling with $N{=}9{,}216$ visual tokens.
- **Appendix E: Micro-Architectural Profiling.** Pruning-overhead and memory profiling up to $N{=}16{,}384$ visual tokens.
- **Appendix F: Algorithmic Derivations and Complexity Analysis.** Primal–dual derivation of SG-RLS and complexity analysis.
- **Appendix G: Directional Suppression Details.** Numerical and implementation details for ranking-based directional suppression.
- **Appendix H: Hyperparameter Robustness.** Sensitivity analysis over α, T, and λ.

A. Experimental Setup

This appendix consolidates models, benchmarks, baselines, and implementation details deferred from Section 3.

A.1. Models

We evaluate SFPruner across three representative MLLM architectures, chosen to span distinct token regimes:

- **LLaVA-NeXT-7B** (Liu et al., 2024b) processes high-resolution images via the dynamic AnyRes strategy, dividing each image into up to five patches and yielding $5\times576{=}2{,}880$ visual tokens. This setting represents *multi-patch high-resolution* encoding.
- **Qwen2.5-VL-7B** (Bai et al., 2025) encodes the entire image as a single continuous token sequence rather than independent patches. This setting represents *dense single-stream* encoding, where all tokens compete within one attention window.
- **LLaVA-Video-7B** (Zhang et al., 2024c) extends MLLM inference to multi-frame video inputs, where temporal stacking further inflates token counts. This setting represents the *temporal multi-frame* regime.

Together these cover image-centric and temporal multimodal settings, allowing us to assess SFPruner under different visual-token budget regimes.

A.2. Benchmarks

Image understanding. We evaluate on a broad suite of vision-language benchmarks: VQA^{v2} (Goyal et al., 2017), GQA (Hudson & Manning, 2019), ScienceQA-IMG (SQA^{IMG}) (Lu et al., 2022), TextVQA (Singh et al., 2019), POPE, MME (Yin et al., 2024), MMBench (MMB), MMBench-CN (MMB-CN), MMVet, AI2D, HallusionBench (HBench), and MMStar.

Video understanding. For video reasoning, we use Video-MME (Fu et al., 2025) (Short / Medium / Long splits), MLVU (Zhou et al., 2024), LongVideoBench (Wu et al., 2024) (Val, Perception, Relation tracks), and MMVBench.

These benchmarks collectively probe fine-grained reasoning, instruction alignment, OCR-style text grounding, hallucination resistance, and large-scale visual token processing.

A.3. Baselines

We compare SFPruner against representative methods from both pruning paradigms:

- **Heuristic methods:** SparseVLM (Zhang et al., 2025b), PruMerge / PruMerge+ (Shang et al., 2025), VisionZip (Yang et al., 2025), and VisPruner (Zhang et al., 2024a).
- **Subset-optimization methods:** DivPrune (Alvar et al., 2025), CDPruner (Zhang et al., 2025a), and D^2Pruner (Zhang et al., 2026).

A.4. Evaluation Metrics

We report **absolute task accuracy** on each benchmark and **relative performance retention** (Rel. %) with respect to the corresponding unpruned vanilla model of the same backbone, defined as the per-task average of pruned-to-vanilla score ratios.

To evaluate computational behavior, we report **selection latency** (ms), measuring only the execution time of the token-selection module before the LLM forward pass. This isolates the algorithmic overhead of pruning from the rest of the pipeline. All latency measurements are performed on a single NVIDIA RTX 4090 GPU. Each measurement is bracketed by `torch.cuda.synchronize()` to capture true kernel execution durations rather than asynchronous host-side timings. Reported values exclude cold load and data I/O; weights are cached in GPU memory before measurement. For peak memory profiling (Appendix E), we additionally invoke `gc.collect()`, `torch.cuda.empty_cache()`, and `torch.cuda.reset_peak_memory_stats()` before each run, so the reported peak corresponds to the transient footprint of internal pruning tensors.

A.5. Implementation Details

Hyperparameters. Unless otherwise noted, we use shared values across all benchmarks: fusion weight α=0.5, softmax temperature T=20, and ridge regularization λ=0.005. For Qwen2.5-VL, which lacks a canonical CLS token, we set α=0 and rely solely on the visual saliency proxy aggregated over spatial self-attention. A complete sensitivity analysis over α, T, and λ is provided in Appendix H.

Computation. The ridge leverage computation in SG-RLS is performed in either the primal ($D\times D$) or dual ($N\times N$) space using the standard primal–dual ridge identity (Appendix F); we choose the smaller system for the Cholesky solve. The pairwise similarity matrix C_{sim} is materialized in FP16 to reduce memory at high N. All operations are implemented in standard PyTorch using batched GEMMs, element-wise comparisons, and row-wise reductions; no custom CUDA kernels are required.

Token-budget conventions. For LLaVA-NeXT, we report per-patch budgets of 128, 64, and 32 retained tokens, corresponding to total budgets of 640, 320, and 160 tokens across five patches (∼22%, ∼11%, and ∼5.5% retention). For Qwen2.5-VL, we report 40% and 20% retention ratios over the dense visual-token sequence. For LLaVA-Video, we report 512 and 256 tokens per frame.

B. Full Image-Benchmark Results and Protocol

B.1. Supplementary Experimental Protocol

We evaluate SFPruner across three model families: LLaVA-NeXT-7B for high-resolution multi-patch images, Qwen2.5-VL-7B for dense single-sequence image encoding, and LLaVA-Video-7B for multi-frame video inference. Image benchmarks cover VQA^{v2}, GQA, SQA$^{\text{IMG}}$, TextVQA, POPE, MME, MMB, MMB$^{\text{CN}}$, MMVet, AI2D, HallusionBench, and MMStar depending on the backbone; video benchmarks cover Video-MME, MLVU, LongVideoBench, and MMVBench. Task scores are reported in their native units and summarized by relative performance retention (Rel. %) against the unpruned vanilla model. Selection latency isolates only the pruning-forward module before the LLM forward pass and is measured on a single NVIDIA RTX 4090 with `torch.cuda.synchronize()` around the timed region.

B.2. LLaVA-NeXT-7B: High-Resolution Multi-Patch Setting

LLaVA-NeXT-7B evaluates SFPruner in the high-resolution multi-patch setting. Table A presents the full sweep at 5×128, 5×64, and 5×32 retained tokens, evaluated on nine image benchmarks: VQA^{v2}, GQA, SQA$^{\text{IMG}}$, TextVQA, POPE, MME, MMB, MMB$^{\text{CN}}$, and MMVet. Relative performance (**Rel.**) is normalized to the unpruned vanilla model. Heuristic-based methods are listed first within each block, followed by subset-optimization methods.

Selection efficiency. Optimization-based methods such as DivPrune and CDPruner rely on iterative subset construction, and their selection latency scales noticeably with the retention budget. At 5×128, DivPrune and CDPruner require 33.8 ms and 35.2 ms, respectively. SFPruner, in contrast, performs covariance-level attenuation and directional masking via parallel tensor operations; its selection latency stays *nearly constant* across budgets (3.4–3.5 ms), as the absence of K-dependent loops removes the iterative bottleneck.

Performance retention. At 5×128 (∼22% retention), SFPruner reaches 100.0% Rel., matching the vanilla baseline within evaluation variance. At 5×64 (∼11%), Rel. drops only to 98.2%, remaining competitive with optimizer-based baselines while requiring substantially lower selection time. Under aggressive compression at 5×32 (∼5.5%), SFPruner retains **96.4%** Rel., outperforming heuristics such as VisionZip (90.0%) and remaining competitive with CDPruner (95.5%) at 3.4 ms vs. 8.9 ms. The trend confirms that single-forward redundancy modeling preserves reasoning quality without sequential subset construction.

B.3. Qwen2.5-VL-7B: High-Resolution Single Sequence

Qwen2.5-VL encodes each high-resolution image as a dense continuous visual-token sequence, creating a stronger single-window redundancy problem than the patch-separated LLaVA-NeXT setting. Table B reports the full result sweep at 40% and 20% retention, matching the main-paper protocol in Table 1.

Table A. **Full results on LLaVA-NeXT-7B** at 5×128, 5×64, and 5×32 retained tokens. **Rel. (%)** is normalized to the unpruned vanilla model; **Time (ms)** is selection latency on RTX 4090 with CUDA synchronization.

Method	VQA^{v2}	GQA	SQA^{IMG}	TextVQA	POPE	MME	MMB	MMB^{CN}	MMVet	Rel. (%)	Time (ms) ↓
Vanilla 5 × 576 *tokens*											
LLaVA-NeXT-7B	81.3	62.5	67.5	60.3	86.8	1511.8	65.8	57.3	40.0	100.0	–
Retain 5 × 128 *tokens*											
SparseVLM (Zhang et al., 2025b)	79.2	61.2	67.6	59.7	85.3	1456.8	65.9	**58.6**	36.1	97.9	19.0
PruMerge+ (Shang et al., 2025)	78.2	60.8	67.8	54.9	85.3	1480.2	64.6	57.3	32.7	98.3	1.2
VisionZip (Yang et al., 2025)	79.1	61.2	**68.1**	**59.9**	86.0	1493.4	65.8	58.1	38.9	99.5	1.1
DivPrune (Alvar et al., 2025)	79.3	61.9	67.8	57.0	86.9	1469.7	65.8	57.3	38.0	98.5	33.8
CDPruner (Zhang et al., 2025a)	79.9	62.6	67.9	58.4	**87.3**	1474.2	66.3	57.5	**41.9**	99.9	35.2
SFPruner	**80.1**	**62.7**	67.6	59.0	87.1	**1480.3**	**67.1**	57.6	41.0	**100.0**	**3.5**
Retain 5 × 64 *tokens*											
SparseVLM (Zhang et al., 2025b)	74.6	57.9	67.2	56.5	76.9	1386.1	63.1	**56.7**	32.8	93.3	18.2
PruMerge+ (Shang et al., 2025)	75.3	58.8	**68.1**	54.0	79.5	1444.2	63.0	55.6	31.4	95.7	1.2
VisionZip (Yang et al., 2025)	76.2	58.9	67.5	**58.8**	82.3	1397.1	63.3	55.6	35.8	95.7	1.1
DivPrune (Alvar et al., 2025)	77.2	61.1	67.7	56.2	84.7	1423.3	63.9	55.7	34.8	96.0	11.9
CDPruner (Zhang et al., 2025a)	78.4	**61.6**	67.8	57.4	**87.2**	1453.1	**65.5**	55.7	**37.9**	97.6	19.2
SFPruner	**78.5**	61.5	67.8	58.3	86.7	**1507.6**	65.1	56.6	**37.9**	**98.2**	**3.4**
Retain 5 × 32 *tokens*											
PruMerge (Shang et al., 2025)	70.5	56.2	66.9	50.3	71.1	1289.6	58.0	48.9	29.3	87.7	1.1
VisionZip (Yang et al., 2025)	71.4	55.2	**67.9**	55.0	74.9	1327.8	58.6	50.4	32.3	90.0	1.1
DivPrune (Alvar et al., 2025)	75.0	59.3	67.1	54.1	80.0	1356.6	62.9	53.7	32.0	92.9	7.8
CDPruner (Zhang et al., 2025a)	**76.7**	**60.8**	67.5	55.4	**86.8**	1425.1	64.2	53.8	36.2	95.5	8.9
SFPruner	76.6	60.3	**67.9**	**57.0**	86.6	**1435.3**	**64.3**	**55.0**	**37.2**	**96.4**	**3.4**

Table B. **Full results on Qwen2.5-VL-7B** at 40% and 20% retention. Rel. (%) is normalized to the unpruned Qwen2.5-VL-7B baseline; **Time (ms)** is pruning-forward selection latency.

Method	TextVQA	AI2D	HBench	MME	MMB	MMB^{CN}	POPE	MMStar	Rel. (%)	Time (ms) ↓
Vanilla 100% tokens										
Qwen2.5-VL-7B	85.3	80.8	64.3	2316.0	79.7	81.8	86.4	56.7	100.0	–
Retain 40% tokens (avg. 512 tokens)										
PruMerge (Shang et al., 2025)	82.1	79.8	62.6	2287.0	78.4	80.0	86.0	55.6	98.1	1.2
VisPruner (Zhang et al., 2024a)	82.6	79.9	61.7	2297.5	78.7	79.9	85.4	55.3	98.0	12.3
VisionZip (Yang et al., 2025)	82.6	79.5	61.2	2306.5	78.3	**80.5**	86.1	55.2	98.0	**1.1**
DivPrune (Alvar et al., 2025)	82.3	78.6	61.4	2230.6	78.5	80.0	84.5	54.7	97.0	23.1
CDPruner (Zhang et al., 2025a)	83.7	79.6	63.4	2302.2	**79.2**	80.3	86.1	**55.9**	98.8	112.4
D^2Pruner (Zhang et al., 2026)	83.2	**82.9**	**63.7**	2305.3	78.2	79.6	85.3	54.9	98.7	31.2
SFPruner	**84.0**	79.9	63.1	**2319.3**	79.0	79.7	**86.5**	**55.9**	**99.0**	2.5
Retain 20% tokens (avg. 256 tokens)										
PruMerge (Shang et al., 2025)	74.7	77.2	59.1	2280.1	77.6	77.7	84.5	52.8	94.6	**1.1**
VisPruner (Zhang et al., 2024a)	75.3	77.4	59.0	2213.9	77.5	78.3	84.5	52.2	94.3	7.5
VisionZip (Yang et al., 2025)	75.9	77.6	59.1	2276.5	77.4	78.5	84.5	53.5	95.1	**1.1**
DivPrune (Alvar et al., 2025)	76.5	76.6	58.1	2203.7	76.9	77.5	83.6	51.8	93.7	11.8
CDPruner (Zhang et al., 2025a)	79.1	78.4	**61.1**	2264.5	77.9	**79.5**	84.9	53.4	96.2	56.7
D^2Pruner (Zhang et al., 2026)	78.6	**79.4**	60.2	2280.5	76.1	78.6	83.0	52.1	95.3	18.9
SFPruner	**79.8**	78.3	60.2	**2295.7**	**78.1**	79.4	**85.7**	**53.7**	**96.5**	2.5

Dense-sequence latency. Optimization-based methods incur visible overhead in this single-sequence regime: at 40% retention, CDPruner, DivPrune, and D^2Pruner require 112.4 ms, 23.1 ms, and 31.2 ms, respectively. At 20% retention, their costs drop with the target subset size, reflecting iterative dependence on K. SFPruner instead remains at 2.5 ms for both budgets because covariance attenuation and directional masking are evaluated in parallel.

Performance retention. At 40% retention, SFPruner reaches 99.0% Rel., matching or exceeding slower optimizers while being much faster. At 20% retention, it retains 96.5% Rel., remaining competitive with CDPruner and D^2Pruner but avoiding their sequential search overhead. These results show that the non-iterative formulation scales to long continuous visual sequences as well as multi-patch image inputs.

B.4. Core Ablation on LLaVA-NeXT-7B

We also include the full component ablation underlying Table 3. All variants retain 64 tokens per patch on LLaVA-NeXT-7B, corresponding to 320 total retained visual tokens across five patches, and report GQA, TextVQA, POPE, aggregate Rel. (%), and pruning-forward selection time.

Table C. **Core ablation on scoring metrics and selection strategies.** SG uses only the semantic guidance score; SG-RLS adds covariance-level leverage attenuation.

Scoring Metric	Selection Strategy	GQA	TextVQA	POPE	Rel. (%)	Time (ms) ↓
SG ($\tilde{S}_{\text{guide}}$)	Top-K Selection	60.3	57.8	85.5	96.9	**0.7**
SG ($\tilde{S}_{\text{guide}}$)	Sequential Search	61.2	58.2	86.3	98.0	18.6
SG ($\tilde{S}_{\text{guide}}$)	Directional Masking	61.2	**58.3**	86.4	97.9	0.8
SG-RLS	**Directional Masking**	**61.5**	**58.3**	**86.7**	**98.3**	2.3

Directional masking vs. sequential selection. Naive Top-K is fastest but leaves spatial redundancy unresolved, yielding 96.9% Rel. Sequential search raises Rel. to 98.0% by updating token state step by step, but its iterative loop costs 18.6 ms. Directional masking achieves similar retention (97.9%) in 0.8 ms by suppressing lower-ranked neighbors in parallel, approximating much of the benefit of iterative exclusion without sequential state updates.

Effect of covariance-level attenuation. Replacing SG with SG-RLS under the same directional mask raises Rel. to 98.3%, with consistent gains on GQA and POPE. The leverage computation adds modest overhead, but the final 2.3 ms latency remains far below sequential selection.

C. Full Video-Benchmark Results: LLaVA-Video-7B

We further evaluate scalability under multi-frame video settings with LLaVA-Video-7B. Video inference inflates token counts via temporal stacking, making redundancy modeling more demanding than in single-image regimes. Table D reports performance and selection latency at 512 and 256 tokens per frame across Video-MME (Short/Medium/Long), MLVU, LongVideoBench (Val/Perception/Relation), and MMVBench.

Latency. Sequential optimizers incur substantial overhead: at 512 tokens/frame, DivPrune and CDPruner spend 131 ms and 92 ms, respectively, while D^2Pruner spends 31.2 ms. SFPruner stays at 3.1 ms, close to lightweight heuristic methods. The latency is also *retention-invariant*: it remains 3.1 ms when reducing the budget from 512 to 256 tokens/frame, since pruning is parallelized.

Performance retention. At 512 tokens/frame, SFPruner attains **94.9%** Rel., the highest among all compared methods. Under stronger compression at 256 tokens/frame, Rel. remains at **92.1%**, comparable to optimizer-based baselines while requiring far less selection time. This shows that the non-iterative formulation transfers cleanly to the temporal regime, converting token reduction into measured wall-clock gains without sacrificing spatio-temporal reasoning.

D. Extreme Scalability: High-Resolution Stress Test

To evaluate structural scalability under extreme conditions, we scale Qwen2.5-VL inputs to a continuous sequence of N=9,216 visual tokens with feature dimension D=3,584 on the full MME dataset. We apply a 30% retention budget ($\sim$2,770 tokens) and profile FLOPs, peak GPU memory, isolated pruning time, end-to-end prefill latency, and total inference time (Table E).

At this scale, the limitation of sequential optimizers becomes acute: CDPruner spends 576 ms and DivPrune spends 458 ms purely for subset selection, eroding the savings from token reduction. SFPruner completes pruning in **28 ms**, reduces prefill latency from 1813 ms (vanilla) to **1114 ms**, and cuts total inference time from 6101 sec to **3601 sec**. This corresponds to a 1.69× end-to-end speedup, whereas DivPrune (5387 sec, 1.13×) and CDPruner (5787 sec, 1.05×) recover far less because their pruning overhead offsets much of the attention savings.

Table D. **Results on LLaVA-Video-7B.** Performance comparison across video benchmarks at 512 and 256 tokens per frame. Rel. (%) is normalized to the unpruned baseline.

Method	Video-MME			MLVU	LongVideoBench			MMVBench	Rel. (%)	Time
	Short	Medium	Long	Avg.	Val	Perception	Relation			(ms) ↓
Vanilla 100% tokens										
LLaVA-Video-7B	67.1	62.8	47.6	60.3	61.1	65.0	53.8	58.8	100.0	–
Retain 512 tokens / frame										
VisPruner (Zhang et al., 2024a)	64.0	50.8	44.0	57.5	53.5	52.5	53.5	56.5	91.0	12.3
VisionZip (Yang et al., 2025)	65.0	51.5	44.5	58.5	54.2	53.0	54.0	58.1	92.4	**1.1**
DivPrune (Alvar et al., 2025)	66.6	**53.0**	45.0	60.1	55.6	**55.2**	55.5	58.1	94.5	131.0
CDPruner (Zhang et al., 2025a)	64.6	52.0	45.7	60.0	54.9	54.9	54.7	57.1	93.5	92.0
D^2Pruner (Zhang et al., 2026)	65.5	52.5	**46.0**	59.5	55.0	54.5	54.8	57.5	93.8	31.2
SFPruner	**67.4**	52.6	**46.0**	**60.2**	**55.7**	**55.2**	**55.6**	**58.3**	**94.9**	3.1
Retain 256 tokens / frame										
VisPruner (Zhang et al., 2024a)	60.5	47.0	41.5	54.0	50.5	49.5	49.5	53.0	85.3	7.5
VisionZip (Yang et al., 2025)	61.5	48.0	42.0	55.0	51.0	50.5	50.0	54.5	86.8	**1.1**
DivPrune (Alvar et al., 2025)	**64.1**	51.2	45.3	**58.3**	**54.1**	**56.3**	52.6	54.9	91.9	64.0
CDPruner (Zhang et al., 2025a)	61.1	49.6	46.0	57.2	52.4	52.9	51.9	52.8	89.4	44.0
D^2Pruner (Zhang et al., 2026)	62.1	49.9	45.4	57.5	52.8	52.0	52.2	55.2	90.0	18.9
SFPruner	**64.1**	**51.8**	**47.0**	58.2	53.1	53.2	**53.0**	**56.7**	**92.1**	3.1

Memory. Peak memory remains comparable across methods. SFPruner consumes 18,432 MB—lower than the unpruned full sequence (19,749 MB) and slightly under the optimizer baselines—confirming that covariance-level redundancy attenuation does not introduce extra memory burden even in ultra-high-resolution settings.

E. Micro-Architectural Profiling and Code-Level Analysis

To clarify the limited speedups of existing baselines at high resolutions, we profile their pruning overhead at the kernel level. The total wall-clock latency can be viewed as

$$T_{\text{total}} = T_{\text{inference}}(K) + T_{\text{pruning}}(N, K),$$

where $T_{\text{inference}}(K)$ decreases with the retained token budget, but $T_{\text{pruning}}(N, K)$ may offset these savings if the selection policy is expensive.

We conduct standalone PyTorch micro-benchmarks at fixed D=3,584 and 30% retention while sweeping N from 256 to 16,384 (Section E.1). Across 10 trials with explicit `torch.cuda.synchronize()` boundaries, the pure selection latency of SFPruner is 27.7 ms at N=9,216 and 54.12 ms at N=16,384. In contrast, DivPrune and CDPruner, which rely on iterative `for i in range(keep_k):` loops with sequential reductions such as `torch.min`, `torch.argmax`, and `torch.einsum`, reach 2,630 ms and 1,435 ms at N=16,384, respectively—large enough to negate much of the inference speedup from token reduction.

Why parallelism wins. SFPruner removes the K-dependent loop entirely. The Ridge Leverage Score is computed via a single batched Cholesky solve, the directional masking step is implemented with parallel tensor operations, and Top-K runs once. The GPU thus evaluates redundancy with a small number of large kernels rather than many sequential update steps, eliminating the kernel-launch and synchronization overhead that dominates iterative pruners.

E.1. Benchmarking Methodology

We isolate the algorithmic overhead of the pruning module from the full LLM pipeline using a standalone PyTorch micro-benchmark on a single NVIDIA RTX 4090. DivPrune and CDPruner are imported from their official GitHub repositories; intermediate visual embeddings are simulated with D=3,584 and 30% retention while N is swept from 256 to 16,384.

- **Latency.** We bracket every measurement with `torch.cuda.synchronize()` to capture true kernel execution durations rather than asynchronous host-side timings.

Table E. **Hardware profiling under extreme scale** on Qwen2.5-VL (N=9,216, full MME). All methods retain 30% tokens; selection latency, prefill latency, and total inference time are reported on a single RTX 4090.

Method	FLOPs (T) ↓	Peak Memory (MB) ↓	Pruning (ms) ↓	Prefill (ms) ↓	Infer Time (sec) ↓
Vanilla 100% tokens					
Full Seq.	148	19,749	–	1813	6101
Retain 30% tokens					
DivPrune (Alvar et al., 2025)	44.4	18,512	458	1547	5387
CDPruner (Zhang et al., 2025a)	44.4	18,736	576	1658	5787
SFPruner	44.4	**18,432**	**28**	**1114**	**3601**

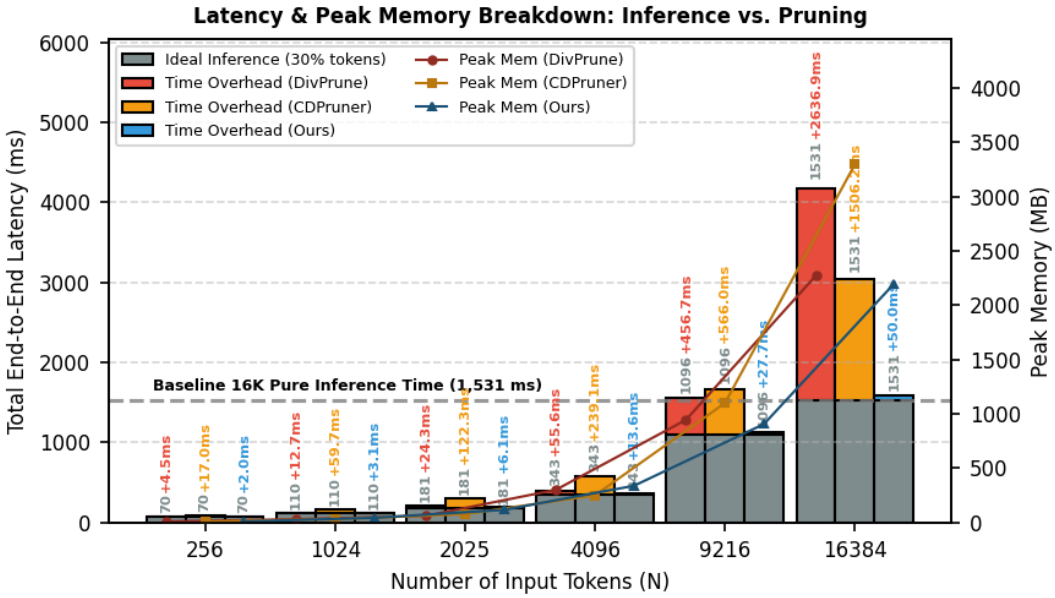

Figure A. **End-to-end latency and peak memory across token counts N.** Sequential optimizers (DivPrune, CDPruner) accumulate severe overhead as N grows, while SFPruner maintains low parallelized selection overhead up to N=16,384, closely approaching the pure inference time of the 16K baseline (1,531 ms). Bar charts: total latency. Line plots: peak memory.

- **Peak memory.** Before each measurement, we invoke `gc.collect()` and `torch.cuda.empty_cache()`, then call `torch.cuda.reset_peak_memory_stats()`, so the reported peak corresponds to the transient footprint of internal tensors.

E.2. Latency Analysis: Sequential Overhead

The latency growth of subset-optimization methods at high resolution stems from the sequential nature of their selection procedure. Each iteration depends on the result of the previous one; reductions and state updates over the remaining tokens cannot be fused across iterations on the GPU. For target subset size K, this produces $\mathcal{O}(K)$ kernel-launch and synchronization rounds.

In contrast, SFPruner expresses redundancy evaluation as a small set of large matrix operations, element-wise comparisons, and reductions. Although the pairwise masking stage still materializes an $N \times N$ similarity matrix, its computation is parallelized rather than sequentially updated over the retained subset. This explains why SFPruner remains substantially faster than iterative subset optimizers in the high-N regime shown in Figure A.

E.3. Peak Memory Analysis

Sequential optimizers also accumulate intermediate state. CDPruner, for example, instantiates a 3D historical state tensor of shape (K, B, N) to support iterative orthogonalization, which scales quadratically with N once K scales linearly with N;

we measure 3,296 MB at the 16K scale. DivPrune slices intermediate distance buffers, such as `cosine_dist[s[:i]]`, at every step, reaching 2,272 MB.

SFPruner bounds peak memory to 2,195 MB by (i) computing ridge leverage in the smaller primal or dual system (Section F), and (ii) materializing the pairwise similarity matrix in FP16, where $N\times N\times 2$ bytes occupies 512 MiB at N=16,384. Within the profiled range, this fits comfortably in standard GPU VRAM, requiring no chunking, tiling, or sparse operators.

F. Algorithmic Derivations and Complexity Analysis

Section 2.2 forms SG-RLS from diagonal leverage scores ℓ_i of the ridge hat matrix $\mathbf{H}$. A naïve $N\times N$ Gram inversion is $\mathcal{O}(N^3)$; the standard primal–dual ridge identity gives the same leverage scores from either the primal ($D\times D$) or dual ($N\times N$) system, allowing us to use the smaller solve.

For $V\in\mathbb{R}^{N\times D}$ and $\lambda>0$,

$$\mathbf{H}_{\text{primal}} = V\,(V^\top V+\lambda I_D)^{-1}V^\top, \qquad \mathbf{H}_{\text{dual}} = (VV^\top+\lambda I_N)^{-1}VV^\top. \tag{11}$$

From

$$V(V^\top V+\lambda I_D) = (VV^\top+\lambda I_N)V,$$

we obtain

$$(VV^\top+\lambda I_N)^{-1}V = V(V^\top V+\lambda I_D)^{-1}.$$

Right-multiplying by $V^\top$ yields $\mathbf{H}_{\text{primal}}=\mathbf{H}_{\text{dual}}=\mathbf{H}$, so $\ell_i=[\mathbf{H}]_{i,i}$ is exact in either representation.

Cost. The ridge solve requires $\mathcal{O}(\min(N,D)^3)$ inversion work up to lower-order terms. For $N>D$ (typical high-resolution MLLMs, e.g., N=9,216, D=3,584), the primal form uses $\mathcal{O}(ND^2)$ work to form $V^\top V$ plus $\mathcal{O}(D^3)$ for Cholesky. For $N\le D$, the dual form requires $\mathcal{O}(DN^2+N^3)$ and is preferable when $N^3\ll D^3$. The directional masking stage additionally computes pairwise token similarities and reductions over an $N\times N$ matrix. This term is parallelizable on GPU, but it is not iterative in K; this distinction is what separates SFPruner from sequential subset optimizers in practice.

G. Implementation Details of Directional Suppression

We record algebraic and engineering choices that keep directional suppression numerically well-behaved beyond the main-text definition.

In Section 2.3, the suppression penalty is built from a score-ordered triangular mask and the pairwise similarity matrix C_{sim}. The implementation must ensure that penalties act as *decay* factors on SG-RLS rather than accidentally amplifying scores.

Clamped penalties under negative cosines. Cosine similarity can be negative for inversely correlated features, so evaluating $1-\max_j(M_{ij}\cdot C_{\text{sim},ij})$ naively can yield $P_i>1$, which would increase $S_{\text{final},i}=S_{\text{SG-RLS},i}\cdot P_i$ instead of suppressing redundancy. We clamp the overlap term before subtracting from 1:

$$P_i = 1-\max\Big(0,\max_j(M_{ij}\cdot C_{\text{sim},ij})\Big), \tag{12}$$

so $P_i\in[0,1]$ and $S_{\text{final},i}\ge 0$. In PyTorch, this corresponds to:

```
sim.max(dim=-1)[0].clamp_(min=0.0)
```

applied to the masked similarity row before multiplying with $S_{\text{SG-RLS}}$.

Max pooling vs. mean or softmax. We use a max reduction rather than mean or softmax to mirror non-maximum suppression: token i is suppressed if it overlaps strongly with at least one higher-priority token. A mean-based penalty can dilute the signal when a redundant token is compared with many orthogonal background tokens, allowing near-duplicates to pass the filter.

H. Hyperparameter Robustness Analysis

We emphasize structural robustness rather than exhaustive per-dataset tuning. SFPruner exposes three knobs—semantic fusion weight α, softmax temperature T (denoted τ in Eq. 1), and ridge regularization λ in Eq. 6. Instead of hand-tuning each benchmark, we sweep wide ranges on five vision-language tasks (MME, SQA, GQA, TextVQA, POPE) and observe narrow Rel. bands.

H.1. Semantic Fusion Weight α

$\alpha \in [0, 1]$ trades off instruction-aware textual relevance against intrinsic visual saliency (Eq. 4). Table F reports Rel. at three LLaVA-NeXT budgets: 128, 64, and 32 retained tokens per patch.

Stable modality fusion. Rel. remains stable for $\alpha \in [0.3, 0.7]$: whenever both modalities contribute without an extreme bias toward text or vision alone, redundant tokens are still identified reliably. The peak shifts mildly with budget—α=0.4 at 128 tokens, α=0.5 at 64 and 32 tokens—yet row-wise variation stays below 1 pp Rel., indicating low sensitivity to minor shifts in modality weighting.

Default. We therefore adopt α=0.5 as a balanced default; fine-grained per-dataset tuning is unnecessary for the compression regimes we study.

H.2. Sensitivity to Temperature (T) and Ridge Regularization (λ)

Table G sweeps T and λ at the 64-token-per-patch budget on LLaVA-NeXT-7B.

Temperature T. T controls the sharpness of the textual-relevance distribution (Eq. 1). Sweeping $T \in \{10, 15, 20, 25, 30\}$ keeps Rel. between 98.6% and 99.2%, showing that token ranking does not hinge on a brittle sharpness setting. We set T=20 as the default.

Ridge constant λ. λ stabilizes the covariance inversion in SG-RLS (Eq. 6). Scaling λ over a 20× range (0.001–0.020) shifts Rel. only between 98.8% and 99.2%, consistent with a well-conditioned ridge formulation where covariance attenuation is stable without exact tuning. We adopt λ=0.005 as the default.

Table F. **Robustness to α** on LLaVA-NeXT-7B at 128/64/32 retained tokens per patch. Per-budget optima are highlighted in bold.

Tokens	α	MME	SQA	GQA	TextVQA	POPE	Rel. (%)
128	0.3	1468	**67.9**	62.3	**59.8**	87.2	99.4
	0.4	1472	**67.9**	62.6	**59.8**	87.2	**99.6**
	0.5	1480	67.6	**62.7**	59.0	87.1	99.3
	0.6	1483	67.6	62.6	59.3	87.2	99.4
	0.7	**1485**	67.8	62.6	59.2	**87.3**	99.5
64	0.3	1488	67.4	61.1	**58.5**	**86.8**	98.6
	0.4	1496	67.7	61.3	58.3	86.7	98.8
	0.5	**1507**	**67.8**	**61.6**	58.3	86.7	**99.1**
	0.6	1495	67.5	61.3	58.2	**86.8**	98.7
	0.7	1487	67.5	61.3	57.8	**86.8**	98.5
32	0.3	**1436**	67.6	60.1	**57.3**	86.3	97.1
	0.4	1426	67.8	60.2	57.1	**86.6**	97.1
	0.5	1435	**67.9**	**60.3**	57.0	**86.6**	**97.3**
	0.6	1435	67.8	60.2	56.9	**86.6**	97.2
	0.7	1416	67.5	60.1	56.8	86.5	96.7

Table G. **Robustness to T and λ** on LLaVA-NeXT-7B at 64 retained tokens per patch. Defaults are T=20, λ=0.005 (highlighted).

Setting	Config	MME	SQA	GQA	TextVQA	POPE	Rel. (%)
Baseline	Full Model	1511.0	67.5	62.5	60.3	86.8	100.0
T	10	1485.0	67.5	61.1	**58.5**	86.7	98.6
	15	1490.6	**67.8**	61.2	58.4	86.7	98.8
	20	**1507.6**	**67.8**	**61.5**	58.3	**87.2**	**99.2**
	25	1503.2	67.6	61.4	58.2	86.7	98.9
	30	1503.0	67.5	61.3	58.4	86.6	98.8
λ	0.001	1502.9	**67.9**	61.4	58.2	86.7	98.9
	0.005	1507.6	67.8	**61.5**	58.3	**87.2**	**99.2**
	0.010	**1510.4**	67.6	61.4	58.0	86.6	98.9
	0.015	1499.9	67.8	61.4	58.2	86.6	98.8
	0.020	1504.7	67.7	61.3	**58.4**	86.6	98.9