

SieveServe: Adaptive Visual Token Pruning for Reliable and Efficient Multimodal LLM Inference

Xiao Li, Xiao Shi, Cheng Zhang, Sa Wang
Zhining Zhang, Xiaofang Zhao

Institute of Computing Technology, Chinese Academy of Sciences

Correspondence: lixiao24s@ict.ac.cn, shixiao@ict.ac.cn

Abstract

Multimodal large language models (MLLMs) increasingly rely on dynamic-resolution visual encoding that produces thousands of tokens per image, creating a scalability bottleneck in both inference latency and concurrent serving capacity. Visual token pruning offers a promising remedy, yet deploying it in practice faces two barriers: existing methods predominantly operate within the LLM’s intermediate layers, violating structural invariants of production inference engines (KV cache granularity, fused kernels, prefix caching); and a uniform pruning ratio causes catastrophic accuracy collapse on dense-information tasks such as document and chart understanding. We present **SieveServe**, to our knowledge the first framework that makes visual token pruning deployable inside production MLLM serving engines while providing per-request accuracy safety. SieveServe operates entirely before the LLM—the only pruning paradigm compatible with PagedAttention, FlashAttention, and prefix caching—and adapts pruning intensity per request through two lightweight guards: a Query Complexity Router (QCR) that classifies query difficulty and a Dense Information Guard (DIG) that detects information-rich images. Evaluated on Qwen2.5-VL-7B and InternVL3-8B across eight benchmarks, SieveServe retains $\geq 95\%$ baseline accuracy on every benchmark—and $\geq 97.8\%$ on dense-information tasks—recovering over 20 points destroyed by fixed-ratio pruning, while simultaneously reducing P90 TTFT by $5.4\times$ and sustaining $2.2\text{--}2.7\times$ higher throughput under latency SLO constraints.

1 Introduction

The remarkable capabilities of multimodal large language models (MLLMs) have driven their rapid adoption in real-world applications (Liu et al., 2023; Bai et al., 2023; Chen et al., 2024d; Bai

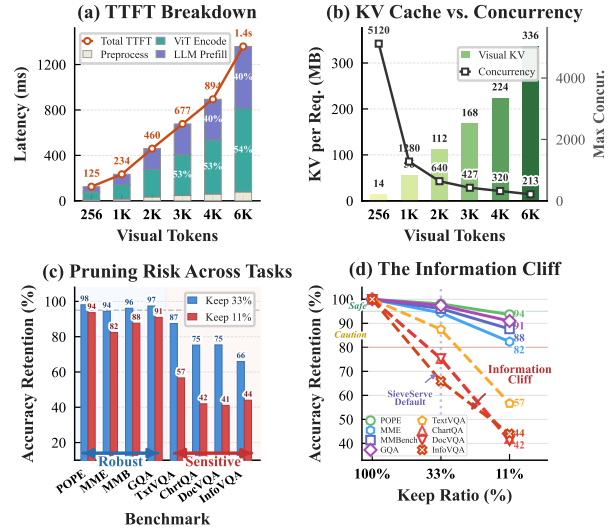


Figure 1: Motivating analysis for SieveServe. (a) Visual tokens dominate TTFT as image length increases. (b) KV-cache footprint grows with visual-token count, limiting concurrent capacity. (c) Fixed-ratio pruning has divergent effects across task types, preserving general visual QA while collapsing dense-information tasks. (d) Dense-information tasks exhibit an information cliff, motivating request-adaptive pruning instead of a uniform ratio.

et al., 2025; Zhu et al., 2025). To improve perceptual fidelity, leading MLLMs such as Qwen2.5-VL and InternVL3 have adopted dynamic-resolution designs that routinely encode a single image into thousands of visual tokens (Wang et al., 2024; Bai et al., 2025; Chen et al., 2024c,b; Xu et al., 2024).

In practice, user-submitted images are comparably demanding: on the WildVision-Chat benchmark (Lu et al., 2024), Qwen2.5-VL produces a median of 1,371 visual tokens per image (9,834 at P90) versus only 31 text tokens, making visual tokens the dominant computational factor (Figure 2). This poses a serious scalability challenge: TTFT grows super-linearly with visual token count, ex-

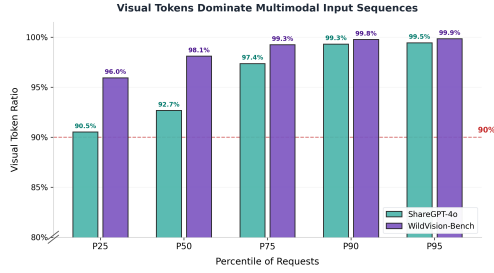


Figure 2: Visual token ratio across request percentiles on two production workloads. Even at P25, visual tokens exceed 90% of total input tokens.

ceeding 1.3s at 6K tokens (Figure 1(a)); meanwhile, per-request KV cache footprint balloons proportionally (Figure 1(b)), reducing maximum concurrency from 70 to fewer than 5 on a single A100-80GB GPU.

A natural response is visual token pruning, and indeed recent work has shown that a large fraction of visual tokens can be removed with minimal accuracy impact (Chen et al., 2024a; Shang et al., 2024; Xing et al., 2024; Wen et al., 2025; Zou et al., 2025; Alvar et al., 2025; Yang et al., 2025b). Yet deploying pruning in production serving reveals two fundamental barriers. The first is **structural incompatibility**: most pruning methods—including FastV and PruMerge (Chen et al., 2024a; Shang et al., 2024)—operate within the LLM’s intermediate layers, dynamically altering sequence lengths per layer. This directly violates the structural invariants of modern inference engines—block-granularity KV cache, fused attention kernels, fixed-sequence batching, and prefix caching (Kwon et al., 2023; Zheng et al., 2024; Yu et al., 2022; Dao et al., 2022; Dao, 2024; Ye et al., 2025)—forcing recent methods to resort to partial workarounds that remain incompatible with full engine optimization (Yang et al., 2025a; Wang et al., 2026). The second barrier is **accuracy unsafety**: retaining 33% of visual tokens preserves over 95% accuracy on natural-image tasks (POPE, MMBench) (Li et al., 2023b; Liu et al., 2024b) yet causes 37.5% collapse on ChartQA (Masry et al., 2022) (Figure 1(c)), with a sharp “information cliff” (Figure 1(d)) making any uniform pruning policy inherently unsafe for mixed workloads.

These barriers call for a rethinking of how pruning integrates with MLLM serving. Our investigation reveals that (1) only pruning performed *before* tokens enter the LLM can satisfy all engine constraints simultaneously, and (2) pruning sensi-

tivity is predictable from two orthogonal signals: query complexity and image information density (Figure 1(c–d)).

Building on these observations, we introduce **SieveServe**, a framework that bridges visual token pruning with production MLLM inference. SieveServe establishes pre-LLM, vision-centric pruning as the only deployment-compatible paradigm and introduces an adaptive per-request policy driven by a Query Complexity Router (QCR) and a Dense Information Guard (DIG), enabling aggressive pruning for the majority of requests while fully preserving information-dense ones. The framework is implemented atop vLLM, spanning three MLLM resolution families.

Our contributions are:

- **Bridging pruning to production serving.** We formalize the structural constraints that make existing pruning methods incompatible with production engines and establish pre-LLM vision-centric pruning as the only deployable paradigm. SieveServe, implemented within vLLM, reduces P90 TTFT by $5.4\times$ and sustains $2.2\text{--}2.7\times$ higher throughput under latency SLO constraints.
- **Accuracy-safe adaptive pruning.** We propose a dual-signal mechanism—QCR for query complexity, DIG for image density—that determines per-request pruning intensity, retaining $\geq 95\%$ accuracy on all benchmarks and $\geq 97.8\%$ on dense-information tasks.

2 Related Work

Multimodal LLMs. Modern MLLMs project ViT-encoded patch embeddings into the LLM’s token space (Liu et al., 2023, 2024a; Li et al., 2023a; Bai et al., 2023; Chen et al., 2024d). Dynamic-resolution designs tile high-resolution images into multiple crops (Wang et al., 2024; Bai et al., 2025; Chen et al., 2024c), producing thousands of visual tokens that dominate compute and memory.

Visual token pruning. Existing pruning methods fall into five paradigms. (i) *LLM intermediate-layer pruning* uses attention to drop tokens (Chen et al., 2024a; Shang et al., 2024; Xing et al., 2024; Yang et al., 2025a) but introduces per-layer token-count variation incompatible with paged caches. (ii) *Encoder-internal pruning* (Rao et al., 2021; Liang et al., 2022; Chen et al., 2023) operates

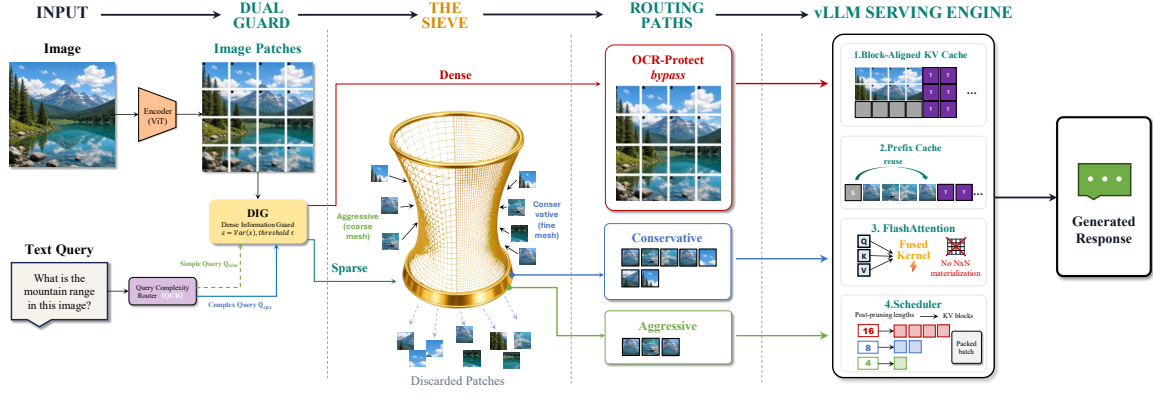


Figure 3: Overview of SieveServe. Given an image–query pair, the Dense Information Guard (DIG) and Query Complexity Router (QCR) jointly determine the pruning path. Dense-information images bypass token pruning and instead receive resolution capping; sparse images are routed through a pluggable visual token sieve at aggressive or conservative intensity. The pruned sequence enters the vLLM serving engine as a block-aligned, fixed-length input compatible with PagedAttention, FlashAttention, and prefix caching.

within the ViT but lacks query signals. (iii) *Pre-LLM vision-centric pruning* selects tokens before the LLM (Bolya et al., 2022; Zou et al., 2025; Alvar et al., 2025; Yang et al., 2025b), keeping sequence structures intact and compatible with serving engines. (iv) *Decode-stage KV eviction* removes entries during decoding without reducing prefill cost (Zhang et al., 2023; Li et al., 2024b; Cai et al., 2024). (v) *Learned pre-LLM scorers* require costly retraining per backbone (Shang et al., 2024; Yang et al., 2025b). All focus on offline evaluation, leaving serving compatibility unaddressed.

MLLM serving infrastructure. Production engines such as vLLM (Kwon et al., 2023), SGLang (Zheng et al., 2024), LMDe-ploy (InternLM Team, 2024), and TensorRT-LLM (NVIDIA, 2024) impose two hard constraints. First, PagedAttention organizes the KV cache into fixed-size blocks with a shared block table, assuming uniform per-layer token counts; intermediate-layer pruning violates this assumption. Second, fused kernels (Dao et al., 2022; Dao, 2024; Ye et al., 2025) never materialize $N \times N$ attention matrices, making attention-score-based pruning infeasible without disabling the kernel and sacrificing 2–4 $\times$ throughput. Pre-LLM vision-centric pruning is the only paradigm that avoids both conflicts.

3 SieveServe

3.1 Overview

SieveServe bridges visual token pruning with production MLLM inference, addressing two complementary challenges: *structural compatibility* with modern serving engines, and *accuracy safety* via per-request adaptive pruning intensity. Given an image–query pair, a lightweight dual-guard policy—combining query-side complexity estimation (QCR) and image-side density detection (DIG)—routes the request to aggressive pruning, conservative pruning, or resolution-capped bypass. All decisions are made before LLM prefill, producing a fixed-length visual sequence indistinguishable from unpruned input. The framework is architecture-agnostic, implemented within vLLM across fixed-resolution (LLaVA-1.5), discrete-tile (InternVL3), and dynamic native-resolution (Qwen2.5-VL) models, with pluggable pruning backends (HoloV, DivPrune, PiToMe). Experiments report results on InternVL3 and Qwen2.5-VL. Figure 3 illustrates the full pipeline.

3.2 Serving-Compatible Pruning

In this section, we identify the structural constraints that production serving engines impose on token pruning and explain why only one pruning paradigm is compatible. Recent pruning methods have begun to acknowledge these constraints (Yang et al., 2025a; Wang et al., 2026); we organize them around three conflict sites.

Pruning Paradigm	Representative Methods	Structural Invariants					Compatible
		Uniform KV	Query-indep.	Decode-transp.	Pre-LLM	FA-compat.	
LLM intermediate-layer	FastV, PruMerge, PyramidDrop	✗	✗	✗	✗	✗	✗
Encoder-internal	DynamicViT, EViT	✓	✓	△	✓	✓	△
Decode-stage KV eviction	H2O, SnapKV	✓	✗	✗	✗	✗	✗
Learned pre-LLM scorers	VisionZip, LLaVA-PruMerge	✓	✓	✓	✓	✓	✗ [†]
Pre-LLM vision-centric	HoloV, DivPrune, PiToMe	✓	✓	✓	✓	✓	✓

Table 1: Serving compatibility of pruning paradigms against five structural invariants from Section 3.2. [†]Learned scorers satisfy all five but require per-backbone retraining.

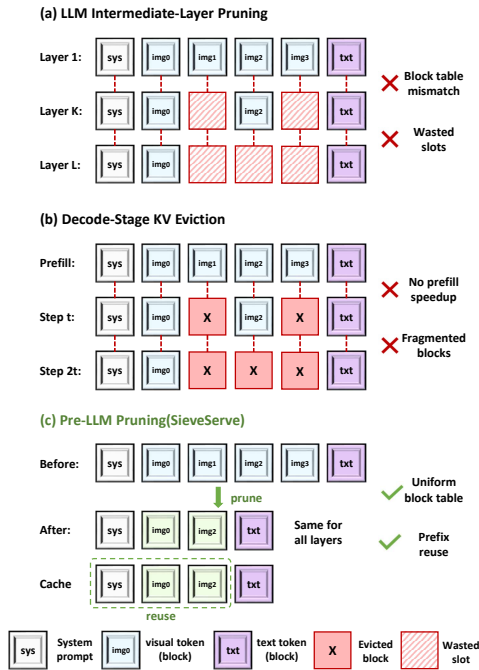


Figure 4: KV cache block behavior under three pruning paradigms. (a) Intermediate-layer pruning creates per-layer block table mismatches. (b) Decode-stage eviction fragments blocks without prefill benefit. (c) Pre-LLM pruning (SieveServe) produces uniform, block-aligned sequences enabling prefix cache reuse.

Conflict 1: Block table integrity. PagedAttention manages the KV cache through a shared block table across all transformer layers (Kwon et al., 2023), requiring uniform per-layer token counts. LLM intermediate-layer pruning (Chen et al., 2024a; Shang et al., 2024; Xing et al., 2024) violates this by removing different tokens at different layers (Figure 4(a)); decode-stage KV eviction (Zhang et al., 2023; Li et al., 2024b) avoids

per-layer mismatches but fragments blocks without reducing prefill cost (Figure 4(b)). SieveServe completes all pruning *before* LLM entry, producing a uniform block table (Figure 4(c)) and enabling the scheduler to operate on the actual post-pruning length.

Conflict 2: Query-dependent scoring. Methods that score visual tokens via LLM cross-attention (Chen et al., 2024a; Wen et al., 2025; Zhang et al., 2024b) couple the pruning outcome to the specific query, making the visual intermediate state non-deterministic and breaking prefix caching (Zheng et al., 2024). SieveServe enforces query-independent pruning: all scoring uses only vision encoder outputs, producing deterministic pruned sequences that enable full prefix cache reuse.

Conflict 3: Attention kernel incompatibility. FlashAttention and FlashInfer (Dao et al., 2022; Dao, 2024; Ye et al., 2025) achieve 2–4 $\times$ throughput without materializing the $N \times N$ score matrix. Pruning methods requiring explicit LLM attention scores must disable these fused kernels, negating latency savings. SieveServe avoids this entirely: all scoring occurs within the vision encoder or through embedding-space statistics; the LLM’s fused kernels are never touched. Table 1 formalizes these three conflicts as five structural invariants; only pre-LLM vision-centric pruning satisfies all five, and SieveServe operationalizes this paradigm within vLLM.

3.3 Adaptive Pruning Policy

Structural compatibility determines *which* paradigm to use but not *how aggressively* to prune. Figure 1(c) shows that retaining 33% of visual tokens preserves 94–98% accuracy on pruning-robust tasks (POPE, MME, GQA, MMBench) while collapsing to 42–75% on

		Fixed Ratio Pruning @66.7%						SieveServe (Ours)			
Model	Benchmark	Base	FastV	DART	HoloV	DivPr.	PiToMe	Score	Ret.%	Keep%	Skip%
Qwen2.5-VL	POPE	87.8	85.6	86.3	85.4	86.7	86.4	86.9	99.0	30.9	0.4
	MME	2337	2245	2190	2190	2213	2190	2239	95.8	37.0	10.2
	GQA	60.8	56.4	58.7	58.6	50.7	50.4	58.9	96.9	32.7	2.2
	MMBench	79.8	75.9	76.8	75.8	71.0	71.3	76.1	95.4	32.1	14.5
	DocVQA	94.9	77.9	91.5	67.8	17.0	16.7	93.9	99.0	98.8	94.0
	InfoVQA	81.7	69.5	53.4	48.0	19.8	17.9	80.7	98.7	97.6	98.9
	ChartQA	86.4	76.7	63.0	54.0	12.7	11.9	84.5	97.8	100.0	100 [†]
	TextVQA	82.9	82.0	72.2	71.3	26.6	26.9	79.9	96.4	47.5	20.4
InternVL3	POPE	91.0	90.6	90.7	90.0	90.8	90.9	90.1	99.0	35.9	7.7
	MME	2403	2378	2318	2243	2273	2267	2372	98.7	52.1	12.4
	GQA	62.4	61.5	62.0	61.4	61.6	61.8	60.5	96.9	33.4	4.1
	MMBench	85.9	80.4	79.5	78.8	79.8	79.9	85.0	98.9	62.6	18.3
	DocVQA	92.2	76.6	76.6	74.0	69.5	65.5	92.1	99.9	99.2	96.1
	InfoVQA	73.9	49.2	47.7	52.6	44.6	45.0	72.7	98.4	96.4	97.5
	ChartQA	86.4	79.5	76.6	75.2	73.3	71.3	86.1	99.7	97.6	96.2
	TextVQA	80.2	78.6	74.8	72.8	72.2	73.6	76.3	95.1	54.6	22.7

Table 2: Accuracy across pruning methods (66.7% ratio) and SieveServe’s adaptive routing. Cell colors: $\geq 95\%$ (SLO met), 80–95% (degraded), $< 80\%$ (collapsed). **Green**: SieveServe. Keep% / Skip%: average kept tokens / DIG bypass rate. SieveServe meets SLO on all 16 tasks; no baseline on more than 9. **Blue**: pruning-robust; **pink**: dense-information. [†]: DIG bypass with resolution cap.

dense-information tasks (ChartQA, DocVQA, InfoVQA); Figure 1(d) reveals a sharp “information cliff” rather than gradual degradation. This divergence reflects the underlying data structure: natural images contain redundant background where pruning is safe, while documents and charts distribute non-redundant information uniformly across patches, making every token potentially irreplaceable. SieveServe addresses this with two complementary signals.

Query Complexity Router (QCR). QCR classifies whether a text query tolerates visual compression. It is implemented as a binary ModernBERT-base classifier (Warner et al., 2025) fine-tuned on 9.2K queries from WildVision-Chat (Lu et al., 2024)—a real-world vision-language arena reflecting production workload distributions—with ground-truth labels obtained via LLM-as-judge annotation. At inference, QCR classifies a query in $\sim 2.5\text{ms}$ with a conservative threshold $\gamma = 0.8$: only high-confidence simple queries are eligible for aggressive pruning, deliberately favoring recall on complex queries to avoid SLO violations.

Dense Information Guard (DIG). DIG detects images whose information density makes pruning unsafe, regardless of query complexity. Given the

patch embedding matrix $X \in \mathbb{R}^{N \times D}$, DIG computes:

$$s_{\text{DIG}}(X) = \frac{1}{D} \sum_{j=1}^D \text{Var}(X_{:,j}). \quad (1)$$

When $s_{\text{DIG}} < \tau$ (default $\tau = 0.55$, calibrated on DocVQA/ChartQA/InfoVQA (Mathew et al., 2021; Masry et al., 2022; Mathew et al., 2022)), token-level pruning is bypassed entirely; instead, SieveServe caps the input resolution to a lower tile count, providing meaningful prefill savings even on dense-information images. DIG is training-free, sub-0.1ms, and computed on already-available encoder outputs.

Dual-guard routing. DIG and QCR interact through three mutually exclusive paths: if DIG flags dense content, pruning is bypassed and resolution is capped; if the image is sparse and QCR classifies the query as simple, aggressive pruning is applied; otherwise, conservative pruning is used. DIG holds absolute veto power, and aggressive pruning requires dual confirmation from both guards—encoding the asymmetry that a missed speedup is preferable to an SLO violation.

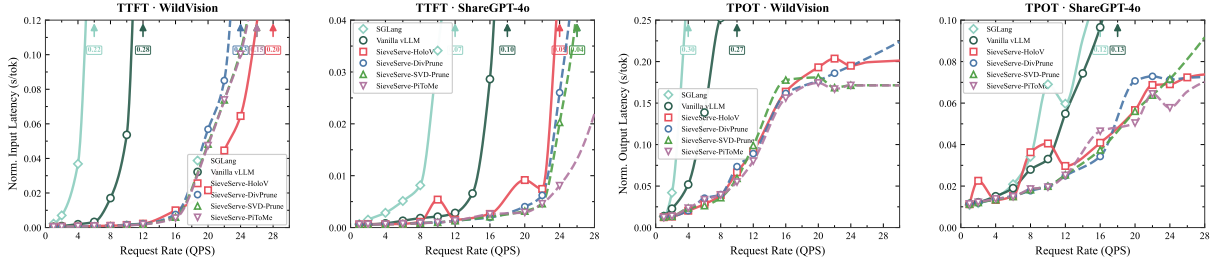


Figure 5: Normalized P90 TTFT and TPOT vs. request rate on WildVision (image-heavy) and ShareGPT-4o (text-heavy) across multiple serving-compatible pruning backends. SieveServe’s latency advantage grows with load due to cascading queue savings, and the same trend holds beyond a single pruning method.

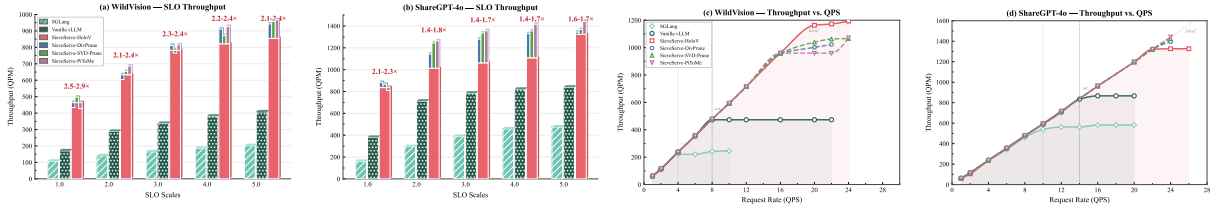


Figure 6: Serving throughput analysis across multiple serving-compatible pruning backends. (a–b) Maximum sustainable throughput under latency SLO constraints: SieveServe achieves 2.2–2.7× throughput while meeting the accuracy SLO. (c–d) Throughput (QPM) vs. request rate: SieveServe sustains higher throughput across load levels, and the same trend holds across HoloV, DivPrune, SVD-Prune, and PiToMe backends.

4 Experiments

We design our evaluation to answer three questions: (1) Does SieveServe’s adaptive routing maintain accuracy across diverse task types? (2) Do the pruning savings translate into real serving gains under production load? (3) What is each component’s individual contribution?

4.1 Setup

We evaluate SieveServe on two representative MLLMs spanning distinct visual tokenization strategies: **Qwen2.5-VL-7B** (Bai et al., 2025) (dynamic native resolution) and **InternVL3-8B** (Zhu et al., 2025) (discrete-tile). We deliberately exclude fixed-resolution models such as LLaVA-1.5 (Liu et al., 2024a) because mainstream MLLM development has converged on dynamic and discrete architectures that better reflect production serving workloads; moreover, fixed-resolution models produce at most ~ 576 tokens, making the visual-token bottleneck far less pronounced. All accuracy experiments run on a single A100-80GB GPU via vLLM (Kwon et al., 2023) with `lmms-eval` (Zhang et al., 2025); serving experiments use $4 \times$ A100 (DP=4, TP=1). We compare against five pruning baselines at 66.7% pruning rate: FastV (Chen et al., 2024a), DART (Wen et al., 2025), HoloV (Zou et al., 2025), DivPrune (Alvar

et al., 2025), and PiToMe (Tran et al., 2024). Eight benchmarks span the pruning-sensitivity spectrum: pruning-robust (POPE, MME, GQA, MMBench) (Li et al., 2023b; Fu et al., 2023; Hudson and Manning, 2019; Liu et al., 2024b) and dense-information (DocVQA, InfoVQA, ChartQA, TextVQA) (Mathew et al., 2021, 2022; Masry et al., 2022; Singh et al., 2019). Serving workloads are WildVision-Chat (Lu et al., 2024) (image-heavy) and ShareGPT-4o (ShareGPT-4o Team, 2024) (text-heavy). We define the **accuracy SLO** as $\geq 95\%$ baseline retention: responses below this threshold are considered unreliable for production deployment, as users cannot distinguish pruning artifacts from genuine model errors.

4.2 Accuracy Under Adaptive Routing

In this section, we investigate whether a single fixed pruning ratio can reliably serve diverse visual workloads, and how SieveServe’s dual-guard routing compares. Table 2 reports accuracy across five pruning baselines (all at 66.7% token removal) and SieveServe on eight benchmarks spanning two models. Cell colors distinguish retention severity: **SLO met** ($\geq 95\%$), **degraded** (80–95%), and **collapsed** ($< 80\%$).

We observe a sharp dichotomy across tasks. On robust benchmarks (blue rows), most baselines re-

tain acceptable accuracy (e.g., $\geq 85\%$ on POPE). On dense-information tasks (pink rows), performance collapses: DivPrune and PiToMe fall below 18% on DocVQA and InfoVQA (Qwen2.5-VL), and even DART—the strongest baseline on DocVQA (91.5%)—drops to 53.4% on InfoVQA. While FastV achieves the highest baseline compliance (9 of 16 pairs) by postponing pruning to intermediate LLM layers (allowing early layers to process all details), it still collapses on dense tasks where information is uniform and cannot tolerate fixed-ratio removal. This collapse occurs because documents and charts distribute critical semantic details uniformly across patches, making fixed-ratio token removal inevitably lossy. Thus, a uniform pruning ratio cannot satisfy accuracy SLOs across heterogeneous inputs. The same trend holds at a stronger shared operating point: when fixed-ratio baselines move from 66.7% to 88.9% removal on Qwen2.5-VL dense-information tasks, all methods degrade further (e.g., HoloV on ChartQA: 54.0 $\rightarrow$ 31.4; FastV on DocVQA: 91.5 $\rightarrow$ 60.8; Appendix E). This confirms that 66.7% is a controlled comparison point rather than a method-specific optimum chosen to favor SieveServe.

SieveServe’s adaptive routing addresses this gap directly. By combining DIG and QCR, the system meets the 95% SLO on all 16 model–benchmark pairs, with $\geq 97.8\%$ retention on every dense-information task. We find DIG accurately separates the two regimes: ChartQA bypasses 100% of images, while POPE bypasses only 0.4%. QCR provides orthogonal query-side safety, classifying 24.1% of POPE queries but only 1.1% of ChartQA queries as simple, showing that the guards capture distinct dimensions of pruning risk. This result shows that adaptive, per-request routing achieves a Pareto frontier that no fixed ratio can reach, and that both guards contribute complementary and non-redundant signals. This dual-guard mechanism prevents the catastrophic failures of uniform policies by restricting visual token reduction strictly to safe regimes.

The token compression efficiency of SieveServe is shown in the Keep% column of Table 2. On robust tasks (POPE), SieveServe prunes aggressively, keeping only 30.9% (Qwen2.5-VL) and 35.9% (InternVL3) of tokens (matching or exceeding the 33.3% baseline). On dense tasks (ChartQA, DocVQA), DIG protects accuracy by keeping $\geq 97.6\%$ of tokens. For mixed tasks (TextVQA),

it dynamically keeps 47.5% and 54.6% of tokens. This selective preservation ensures that scene-text details are retained for reading comprehension while redundant visual backgrounds are pruned. This confirms that QCR’s aggressive routing safely improves efficiency without sacrificing task accuracy. By mapping these custom keep ratios per dataset, SieveServe successfully bounds the model’s visual token budget based on the difficulty of each request rather than relying on a static, worst-case policy. Detailed QCR/DIG parameters and per-benchmark routing statistics are reported in Appendix C. We also evaluate open-ended quality with a unified judge in Appendix E: SieveServe remains close to vanilla on MM-Vet (63.07 vs. 64.82), while fixed-ratio baselines degrade substantially, especially at 88.9% removal.

4.3 Serving Efficiency

Having established that SieveServe preserves accuracy across all benchmarks, we now evaluate whether the pruning savings translate into concrete serving gains under realistic production conditions. We sweep the request rate from 2 to 32 QPS on a 4 $\times$ A100 cluster (DP=4) and benchmark SieveServe against vanilla vLLM, SGLang, and multiple serving-compatible pruning backends integrated into the same SieveServe/vLLM substrate. Following Liu et al. (2025), we generate requests using a Poisson arrival process at each target rate on two workloads—WildVision-Chat (image-heavy) and ShareGPT-4o (text-heavy)—and run each experiment for sufficient duration to reach steady state.

Latency under load. Figure 5 presents the normalized P90 TTFT and TPOT as a function of request rate. We observe that SieveServe’s latency advantage is modest under light load but grows *super-linearly* as the system approaches saturation: at QPS=2 on WildVision, P90 TTFT drops from 4,598 ms to 855 ms (5.4 $\times$); at QPS=16, vanilla vLLM exceeds 30 s while SieveServe remains below 2 s. This amplification is fundamental—pruning reduces not only each individual prefill, but also the queuing delay for *subsequent* requests, creating a cascading benefit that intensifies under heavy load. The same trend appears across HoloV, DivPrune, SVD-Prune, and PiToMe backends, confirming that the system-level gains come from serving-layer components—adaptive routing, scheduler-visible post-pruning lengths, and block-

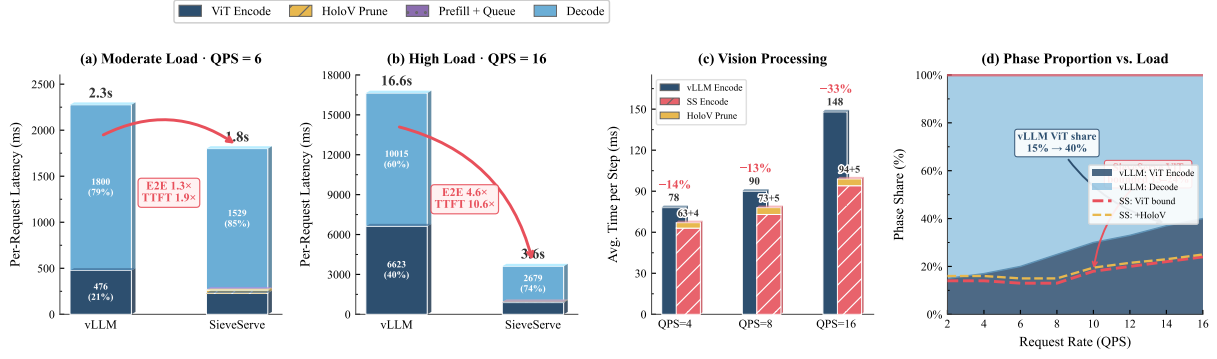


Figure 7: Latency decomposition and vision-processing analysis on WildVision. (a–b) Per-request latency at moderate (QPS=6) and high (QPS=16) load. SieveServe achieves $1.3\times$ and $4.6\times$ E2E speedup, with TTFT amplifying to $10.6\times$ under heavy load. (c) Per-step ViT cost. (d) Phase share vs. load: vanilla vLLM’s ViT share grows from 15% to 40%, while SieveServe bounds it to 13–24%.

aligned execution—rather than a HoloV-specific implementation.

Latency decomposition. To understand where the savings originate, we decompose per-request latency into ViT encoding, LLM prefill, and decode phases. Figure 7(a–b) shows that at moderate load (QPS=6) SieveServe achieves $1.3\times$ E2E speedup, which compounds to $4.6\times$ E2E and $10.6\times$ TTFT at QPS=16 as cumulative queue relief amplifies per-request savings. We further investigate the structural cause in Figure 7(c–d): vanilla vLLM’s ViT share of total latency rises from 15% to over 40% as load increases, meaning vision processing becomes the dominant bottleneck under heavy traffic. SieveServe bounds this share to 13–24% across all load levels, effectively preventing the vision stage from throttling the entire system.

Throughput under SLO. Figure 6(c–d) shows that vLLM saturates at QPS=8 on WildVision, while SieveServe scales linearly to QPS=24. To evaluate throughput under service constraints, Figure 6(a–b) reports the maximum sustainable throughput at linearly scaled latency SLOs ($1.0\times$ to $5.0\times$ baseline TTFT). Under the strictest SLO ($1.0\times$), SieveServe delivers $2.7\times$ higher throughput on WildVision and $2.2\times$ on ShareGPT-4o—the stronger gains on image-heavy workloads reflecting the larger pruning opportunity. Notably, SieveServe is the only system that simultaneously satisfies both the accuracy SLO ($\geq 95\%$ retention) and the latency SLO; the latency backend sweep in Figure 5 further shows that accuracy safety and serving efficiency are complementary outcomes of the serving architecture rather than artifacts of a single pruning algorithm.

4.4 Component and Workload Controls

We further include targeted controls for the two main deployment questions raised by fixed-ratio pruning: whether SieveServe’s accuracy comes from lower aggregate pruning, and whether its dense-task safety is merely an unstructured bypass.

Component ablation. Figure 8 replaces the earlier ablation with the corrected four-setting Cap/DIG control. Fixed HoloV alone is not a deployable serving policy: although it lowers average compute, long visual requests still enter the engine with unsafe resource pressure and trigger 62 OOM fallbacks. The 4K cap removes this tail failure mode, while DIG adds a separate accuracy-risk gate on top of the cap. Thus, the cap and DIG should not be conflated: one enforces resource stability, the other decides when token-level pruning is unsafe.

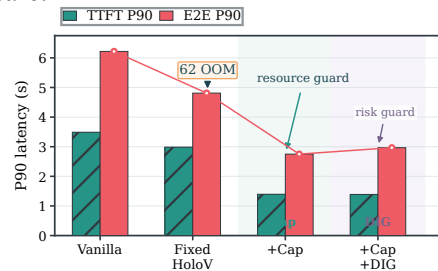


Figure 8: Corrected component ablation on WildVision at QPS=2. Fixed pruning triggers OOM fallbacks; the 4K cap removes this tail/resource failure mode, while DIG adds risk-aware dense-information routing.

Matched-budget traces. Figure 9 tests whether SieveServe is accurate merely because it prunes less. In TraceA, matched HoloV uses the same token-weighted global removal as SieveServe (39.592%), yet SieveServe improves overall accuracy by 6.81 points and high-risk Pool A accu-

racy by 13.12 points. In TraceB, the arrival rate is fixed while the workload shifts to high-risk requests; fixed-ratio methods keep a static budget, whereas SieveServe lowers removal from 55.85% to 24.04%, revealing request-level budget allocation under an accuracy SLO.

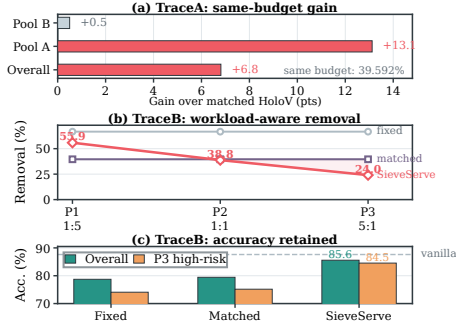


Figure 9: Matched-budget workload controls. (a) Under the same TraceA global removal ratio (39.592%), SieveServe improves especially on high-risk Pool A. (b) In TraceB, SieveServe lowers removal as high-risk requests dominate, showing request-level budget allocation. (c) Under the same serving workload, this dynamic allocation preserves much more overall and high-risk accuracy.

DIG proxy validation and online cost. Figure 10 validates DIG as a lightweight risk proxy rather than an arbitrary bypass rule. Dense-information datasets occupy lower embedding-variance ranges and receive conservative dense routes across two backbones, while robust datasets such as POPE and GQA are rarely bypassed. We choose this training-free proxy because the online path is already competing for GPU resources among serving scheduling, pruning computation, QCR metadata propagation, and KV-cache management. Replacing DIG with an additional learned image predictor would add GPU work precisely when load is high: under identical route decisions, live DIG reduces TTFT P90 from 8406.5 to 6719.5 ms and E2E P90 from 16283.5 to 12207.4 ms at QPS=4. Thus, variance is not claimed as a semantic theory by itself; it is a low-cost, conservative serving proxy that preserves accuracy where pruning risk is high. Together, the matched-budget traces and DIG-cost control clarify the intended operating point: SieveServe is not a lower aggregate pruning ratio disguised as safety, but a request-level budget allocation policy that spends tokens on risky inputs and reclaims them from robust ones while keeping the online decision path cheap enough for serving.

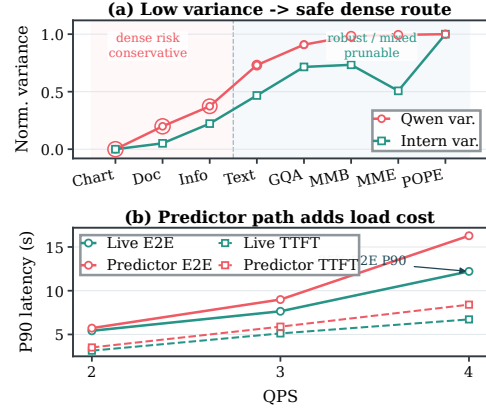


Figure 10: DIG validation and online cost. (a) Low variance identifies dense-information risk and routes such inputs conservatively, while high-variance robust tasks remain prunable. (b) Live DIG avoids the load-amplified overhead that would be introduced by a replayed learned-predictor path.

5 Conclusion

Visual token pruning can relieve the MLLM serving bottleneck, but deployment requires both engine compatibility and accuracy safety. SieveServe provides them through pre-LLM, engine-compatible pruning plus QCR/DIG guards that assign each request the most aggressive safe budget. Across benchmarks, it retains $\geq 95\%$ baseline accuracy ($\geq 97.8\%$ on dense-information tasks), compresses pruning-robust requests by $3.2\times$, reduces P90 TTFT by $5.4\times$, and sustains $2.2\text{--}2.7\times$ higher throughput under latency SLOs, showing that serving-aware pruning can jointly deliver reliability and efficiency. More broadly, our results suggest that visual-token pruning should be treated as a serving policy as much as a token-selection algorithm: the runtime must expose post-pruning lengths, route risk, and block boundaries to the scheduler so that algorithmic compression translates into queue-level gains.

Limitations

Calibration and modality scope. SieveServe’s DIG threshold and QCR classifier are calibrated on English-language, general-purpose benchmarks; different vision encoders, model scales, languages, or specialized domains (e.g., medical imaging) may require re-calibration. The current framework targets single-image, single-turn inputs; multi-turn cache reuse, interleaved multi-image conversations, video understanding, and tasks requiring pixel-level spatial precision (e.g., visual grounding) remain open directions. This scope choice

isolates pruning safety from conversation-level cache placement: in a multi-turn setting, the scheduler must additionally decide whether later turns should be routed to GPUs or nodes that already hold the corresponding image embeddings or visual KV blocks. Our implementation exposes the relevant hooks, including post-pruning token counts, visual-token block boundaries, L0 embedding reuse, and L2 rescue metadata, but does not yet provide a complete session-aware placement policy. We view this as a natural extension of SieveServe rather than a contradiction of the current results: the paper establishes an engine-compatible pruning substrate, while multi-turn deployment requires an additional cache-aware request scheduler on top of that substrate.

Quality and dense-information tradeoffs.

SieveServe inherits a limitation shared by visual token pruning methods: even when benchmark accuracy is preserved, pruned models can generate shorter or less detailed responses than unpruned baselines (Chen et al., 2024a; Wen et al., 2025). Moreover, DIG currently protects document and chart inputs by bypassing token-level pruning, preserving accuracy but forgoing potential speedups on dense-information workloads. Future OCR-aware or length-aware pruning policies may reduce these tradeoffs, but our current system prioritizes conservative serving reliability over aggressive compression in high-risk cases. Accordingly, SieveServe should not be read as claiming that every dense-information request can be safely compressed by the present token scorer. Its guarantee is serving-level: when the runtime detects high pruning risk, it chooses a safer route instead of forcing a fixed deletion ratio that would violate the accuracy SLO.

Ethical Considerations

SieveServe is built on vLLM (Apache-2.0) and evaluated with publicly released models (Qwen2.5-VL-7B, Apache-2.0; InternVL3-8B, MIT) and publicly available benchmarks. The human annotation used for the QCR audit is limited to query-level complexity judgments for research validation; annotators do not provide personal data, and the annotation task does not involve sensitive user information. We will release our code under an open-source license to support reproducibility and further research on serving-compatible visual token pruning.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable feedback and constructive comments. This research is supported by the National Natural Science Foundation of China (NSFC) Joint Fund Project (U24B6012), the Innovation Projects of the Institute of Computing Technology, Chinese Academy of Sciences (E461040, E561080), the Science and Technology Project of the Ministry of Agriculture and Rural Affairs, and the Natural Science Foundation of Beijing (L251082). The AI-driven experiments, simulations, and model training were performed on the robotic AI-Scientist platform of the Chinese Academy of Sciences.

References

- Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. 2025. [DivPrune: Diversity-based visual token pruning for large multimodal models](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9392–9401.
- Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. [Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond](#). *arXiv preprint arXiv:2308.12966*.
- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. [Qwen2.5-VL technical report](#). *arXiv preprint arXiv:2502.13923*.
- Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. [Token merging: Your ViT but faster](#). *arXiv preprint arXiv:2210.09461*.
- Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. 2024. [PyramidKV: Dynamic KV cache compression based on pyramidal information funneling](#). *arXiv preprint arXiv:2406.02069*.
- Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024a. [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*.

- Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, Lixin Gu, Xuehui Wang, Qingyun Li, Yimin Ren, Ping Chen, Jiahao Cai, Hongsheng Xu, Bin Wang, Zhiwei Liu, and 2 others. 2024b. [Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling](#). *arXiv preprint arXiv:2412.05271*.
- Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, Ji Ma, Jiaqi Wang, Xiaoyi Dong, Hang Yan, He Guo, Conghui He, Botian Shi, Zhaohui Jin, Chao Xu, and 10 others. 2024c. [InternVL 1.5: Exploring advanced training and test-time recipes for open-source multimodal models](#). *arXiv preprint arXiv:2404.16821*.
- Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. 2024d. [InternVL: Scaling up vision foundation models and aligning for generic visual-linguistic tasks](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- Zhengsu Chen, Lingxi Xie, Jianwei Niu, Xuefeng Liu, Longhui Wei, and Qi Tian. 2023. [SparseViT: Revisiting activation sparsity for efficient high-resolution vision transformer](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- Tri Dao. 2024. [FlashAttention-2: Faster attention with better parallelism and work partitioning](#). In *International Conference on Learning Representations*.
- Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Re. 2022. [FlashAttention: Fast and memory-efficient exact attention with IO-awareness](#). In *Advances in Neural Information Processing Systems*.
- Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Qiu, Wei Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, and Rongrong Ji. 2023. [MME: A comprehensive evaluation benchmark for multimodal large language models](#). *arXiv preprint arXiv:2306.13394*.
- Drew A. Hudson and Christopher D. Manning. 2019. [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*.
- InternLM Team. 2024. LMDeploy: A toolkit for compressing, deploying, and serving LLMs. <https://github.com/InternLM/lmdeploy>.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. [Efficient memory management for large language model serving with PagedAttention](#). In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023a. [BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models](#). In *International Conference on Machine Learning*.
- Wentong Li, Yuqian Zhang, Jian Liu, Dongqi Xie, Jiahao Jia, and Lei Zhang. 2024a. [TokenPacker: Efficient visual projector for multimodal LLM](#). *arXiv preprint arXiv:2407.02392*.
- Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. 2023b. [Evaluating object hallucination in large vision-language models](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.
- Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Andrea Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024b. [SnapKV: LLM knows what you are looking for before generation](#). *arXiv preprint arXiv:2404.14469*.
- Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. 2022. [EViT: Expediting vision transformers via token reorganizations](#). In *International Conference on Learning Representations*.
- Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024a. [Improved baselines with visual instruction tuning](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. [Visual instruction tuning](#). In *Advances in Neural Information Processing Systems*.
- Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. 2024b. [MMBench: Is your multi-modal model an all-around player?](#) In *European Conference on Computer Vision*.
- Zedong Liu, Shenggan Cheng, Guangming Tan, Yang You, and Dingwen Tao. 2025. [ElasticMM: Efficient multimodal LLMs serving with elastic multimodal parallelism](#). *arXiv preprint arXiv:2507.10069*.
- Yujie Lu, Dongfu Jiang, Wenhui Chen, William Yang Wang, Yejin Choi, and Bill Yuchen Lin. 2024. [Wild-Vision: Evaluating vision-language models in the wild with human preferences](#). In *Advances in Neural Information Processing Systems*.
- Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. [ChartQA: A benchmark for question answering about charts with visual and logical reasoning](#). In *Findings of the Association for Computational Linguistics: ACL*.

- Minesh Mathew, Viraj Bagal, Rubenc Tito, Dimosthenis Karatzas, Ernest Valveny, and C. V. Jawahar. 2022. [InfographicVQA: Part of the DocVQA challenge](#). In *IEEE Winter Conference on Applications of Computer Vision*.
- Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. 2021. [DocVQA: A dataset for VQA on document images](#). In *IEEE Winter Conference on Applications of Computer Vision*.
- NVIDIA. 2024. TensorRT-LLM: Optimizing large language model inference on NVIDIA gpus. <https://github.com/NVIDIA/TensorRT-LLM>.
- Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. 2021. [DynamicViT: Efficient vision transformers with dynamic token sparsification](#). In *Advances in Neural Information Processing Systems*.
- Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. 2024. [LLaVA-PruMerge: Adaptive token reduction for efficient large multimodal models](#). *arXiv preprint arXiv:2403.15388*.
- ShareGPT-4o Team. 2024. ShareGPT-4o: A large-scale multimodal conversation dataset. <https://huggingface.co/datasets/OpenGVLab/ShareGPT-4o>.
- Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Devi Parikh, Marcus Rohrbach, and Dhruv Batra. 2019. [Towards VQA models that can read](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- Hoai-Chau Tran, Duy M. H. Nguyen, Duy M. Nguyen, Trung-Tin Nguyen, Ngan Le, Pengtao Xie, Daniel Sonntag, James Y. Zou, Binh T. Nguyen, and Mathias Niepert. 2024. [Accelerating transformers with spectrum-preserving token merging](#). In *Advances in Neural Information Processing Systems*.
- Haicheng Wang, Yuan Liu, Yikun Liu, Zhemeng Yu, Zhongyin Zhao, Yangxiu You, Zilin Yu, Le Tian, Xiao Zhou, Jie Zhou, Weidi Xie, and Yanfeng Wang. 2026. [POINTS-Long: Adaptive dual-mode visual reasoning in MLLMs](#). *arXiv preprint arXiv:2604.11627*.
- Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024. [Qwen2-VL: Enhancing vision-language model’s perception of the world at any resolution](#). *arXiv preprint arXiv:2409.12191*.
- Yahong Wang, Juncheng Wu, Zhangkai Ni, Longzhen Yang, Yihang Liu, Chengmei Yang, Ying Wen, Lianghua He, Xianfeng Tang, Hui Liu, and Yuyin Zhou. 2025. [When token pruning is worse than random: Understanding visual token information in VLLMs](#). *arXiv preprint arXiv:2512.07580*.
- Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. 2025. [Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2547, Vienna, Austria. Association for Computational Linguistics.
- Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. 2025. [Stop looking for “important tokens” in multimodal language models: Duplication matters more](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 9961–9980.
- Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Ji-qi Wang, Feng Wu, and Dahua Lin. 2024. [Pyramid-Drop: Accelerating your large vision-language models via pyramid visual redundancy reduction](#). *arXiv preprint arXiv:2410.17247*.
- Ruyi Xu, Yuan Yao, Zonghao Guo, Junbo Cui, Zanlin Ni, Chunjiang Ge, Tat-Seng Chua, Zhiyuan Liu, and Gao Huang. 2024. [LLaVA-UHD: An LMM perceiving any aspect ratio and high-resolution images](#). *arXiv preprint arXiv:2403.11703*.
- Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Chendi Li, Jinghua Yan, Yu Bai, Pon-nuswamy Sadayappan, Xia Hu, and Bo Yuan. 2025a. [TopV: Compatible token pruning with inference time optimization for fast and low-memory multimodal vision language model](#). *arXiv preprint arXiv:2503.18278*.
- Lvmin Yang, Yongjie Xu, Jiawei Li, and Zhaoze Chen. 2024. [Fit and prune: Fast and training-free visual token pruning for multi-modal large language models](#). *arXiv preprint arXiv:2409.10197*.
- Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. 2025b. [VisionZip: Longer is better but not necessary in vision language models](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- Zihao Ye, Ruihang Lai, Bo Shao, Tianqi Chen, and Luis Ceze. 2025. [FlashInfer: Efficient and customizable attention engine for LLM inference serving](#). *arXiv preprint arXiv:2501.01005*.
- Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soo-jeong Kim, and Byung-Gon Chun. 2022. [Orca: A distributed serving system for transformer-based generative models](#). In *16th USENIX Symposium on Operating Systems Design and Implementation*.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. 2025. [LMMs-eval: Reality check on the evaluation of large multimodal models](#). In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 881–916, Albuquerque, New Mexico. Association for Computational Linguistics.

Qizhe Zhang, Aosong Cheng, Ming Lu, Zhiyong Zhuo, MinQi Wang, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. 2024a. [Beyond text-visual attention: Exploiting visual cues for effective token pruning in VLMs](#). *arXiv preprint arXiv:2412.01818*.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. 2024b. [SparseVLM: Visual token sparsification for efficient vision-language model inference](#). *arXiv preprint arXiv:2410.04417*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. [H2O: Heavy-hitter oracle for efficient generative inference of large language models](#). In *Advances in Neural Information Processing Systems*.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao, Shiyi Yu, Shiqi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, and 1 others. 2024. [SGLang: Efficient execution of structured language model programs](#). *arXiv preprint arXiv:2312.07104*.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, and 1 others. 2025. [InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models](#). *arXiv preprint arXiv:2504.10479*.

Xin Zou, Di Lu, Yizhou Wang, Yibo Yan, Yuanhuiyi Lyu, Xu Zheng, Linfeng Zhang, and Xuming Hu. 2025. [Don’t just chase “highlighted tokens” in MLLMs: Revisiting visual holistic context retention](#). *arXiv preprint arXiv:2510.02912*.

Appendix

A vLLM Engineering Integration

SieveServe is implemented as a self-contained module (`vllm/v1/logicade/`, 17 files, ~4,000 lines) integrated into vLLM v0.16 at three architecture planes, modifying five vLLM core files (`input_processor.py`, `request.py`, `scheduler.py`, `block_pool.py`, `gpu_model_runner.py`) and one model file (`qwen2_vl.py`). Figure 11 summarizes the integration architecture and data flow.

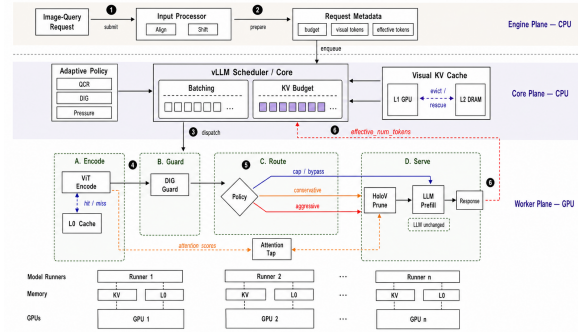


Figure 11: SieveServe integration into vLLM across three architecture planes. Circled numbers indicate the request lifecycle: ①–② block alignment and metadata initialization on CPU, ③–⑤ the ViT→DIG→HoloV pruning pipeline on GPU, ⑥ effective token count feedback to the scheduler.

A.1 Engine Plane (CPU)

Block alignment (`input_processor.py`). Before a request enters the engine, the block alignment module pads the system prompt with new-line tokens (`\n`) until its tokenized length reaches the nearest `block_size` boundary, then shifts all `mm_position` offsets forward by the padding deficit. This ensures visual tokens occupy clean, separable KV blocks—enabling prefix cache reuse and atomic visual block eviction without “dirty blocks” where system-prompt and visual KV entries share the same page.

QCR routing injection (`request.py`). The Query Complexity Router runs *outside* the vLLM engine—as a lightweight API-layer preprocessor using ModernBERT (~400 MB) or a regex fallback—and injects its routing decision into the request via `sampling_params.extra_args`. At construction, the `Request` object extracts three fields: `routing_path` (aggressive/conservative), `bypass_pruning` (QCR protection flag), and `source_type`. The scheduler then propagates these as `scheduled_routing_metadata` to the Worker Plane, where they determine the pruning ratio and DIG bypass logic. This decoupled design keeps the QCR entirely outside vLLM’s critical path—it adds zero latency to the engine and can be independently upgraded or disabled without modifying any vLLM code.

Request metadata (`request.py`). Each request additionally carries pruning-output fields: `pruning_ratio`, `pruned_visual_tokens`, and `effective_num_tokens` (a property returning the

post-pruning token count). These fields propagate pruning results from the Worker Plane back to the Scheduler for accurate KV budget estimation.

A.2 Core Plane (CPU)

The scheduler (`scheduler.py`) is modified in three ways. **(i) Pruning-aware KV budgeting:** The scheduler reads `effective_num_tokens` from request metadata instead of the original visual token count, eliminating overestimation (e.g., 3,000 tokens pruned to 1,000 yields a $3\times$ reduction in KV budget). **(ii) Adaptive pressure controller:** A `DualRatioPruningController` monitors EMA-smoothed KV utilization and issues graduated pressure levels (Normal/Moderate/High/Critical). Under pressure, it triggers L1 visual KV eviction via semantic block ordering (visual blocks before text blocks) but *never* overrides the pruning ratio, preserving accuracy-first semantics. **(iii) L2 DRAM rescue:** When visual KV blocks are evicted from GPU (`block_pool.py`), a callback selectively rescues high-value blocks to pinned host DRAM, enabling sub-millisecond PCIe restore on cache re-hit.

A.3 Worker Plane (GPU)

The GPU-side pipeline in `gpu_model_runner.py` executes three stages per multimodal request:

Stage ③: ViT encode with L0 cache. The model runner first checks the L0 prompt-agnostic embedding cache (LFU eviction, GPU VRAM). On hit, it reuses cached raw embeddings and attention weights, skipping the full ViT encode. On miss, it runs the vision encoder—with a critical modification: we replace the ViT’s last-layer FlashAttention with manual $Q \cdot K^T$ computation (`qwen2_vl.py`) to extract CLS→patch attention weights needed by HoloV, incurring $\sim 5\%$ overhead on the vision encoder only. The LLM backbone and decode phase remain entirely unmodified.

Stage ④: DIG gate. After encoding, the Dense Information Guard computes `embeddings.var(dim=0).mean()` (~ 0.01 ms) and classifies the image as dense-info or sparse. Dense-info images bypass pruning entirely.

Stage ⑤: HoloV pruning. For non-bypassed requests, `apply_pruning_to_encoder_output()` computes a block-aligned keep count, scores tokens using the extracted CLS→patch attention

weights combined with similarity variance, and returns pruned embeddings with `kept_indices`. The post-pruning token count is written to `request.pruned_visual_tokens`, feeding back to the Core Plane via `effective_num_tokens` (⑥).

B Model Selection and Accuracy SLO Justification

Why evaluate on dynamic-resolution models. The majority of foundational visual token pruning methods—FastV (Chen et al., 2024a), PruMerge (Shang et al., 2024), PyramidDrop (Xing et al., 2024), FitPrune (Yang et al., 2024), FasterVLM (Zhang et al., 2024a), VisionZip (Yang et al., 2025b), TokenPacker (Li et al., 2024a)—were developed and primarily evaluated on the LLaVA family (LLaVA-1.5, LLaVA-NeXT), where the visual token count is fixed at 576 or bounded at $\sim 2,880$. While recent work has begun extending evaluation to dynamic-resolution models—DART (Wen et al., 2025) and DivPrune (Alvar et al., 2025) include Qwen2.5-VL results, and TopV (Yang et al., 2025a) tests on InternVL2—production-deployed MLLMs have largely shifted to dynamic-resolution architectures that routinely produce thousands to tens of thousands of visual tokens per image (Bai et al., 2025; Zhu et al., 2025). We evaluate SieveServe on Qwen2.5-VL-7B and InternVL3-8B to align with this deployment reality.

Pruning characteristics differ across architectures. Recent analysis (Wang et al., 2025) observes that different model architectures exhibit different visual token utilization patterns: Qwen2.5-VL utilizes visual tokens in deeper LLM layers compared to LLaVA-1.5, suggesting that pruning behavior may not transfer straightforwardly across architectures. We note this as a relevant observation rather than a definitive conclusion, as controlled cross-architecture comparisons remain limited. What our own experiments do confirm (Table 2) is that, on Qwen2.5-VL and InternVL3 at 66.7% pruning, no fixed-ratio method retains $\geq 95\%$ baseline accuracy across all benchmarks—with several methods falling below 25% retention on dense-information tasks (e.g., DivPrune on DocVQA, PiToMe on InfoVQA).

Why 95% accuracy SLO. We adopt $\geq 95\%$ baseline retention as the accuracy SLO based on the empirical evidence in Table 2: this thresh-

Component	Parameter	Value
QCR	Method	ModernBERT
	Confidence threshold γ	0.80
	Default complexity	complex
DIG	Variance threshold τ	0.55
	Calibration set	Doc/Chart/InfoVQA
Pruning	Conservative ratio	0.667
	Aggressive ratio	0.778
	OCR-protect path	bypass (res. cap)

Table 3: QCR and DIG hyperparameters.

old is strict enough that no fixed-ratio pruning method satisfies it universally across all 16 benchmark-model pairs (the best fixed method meets it on only 9 of 16), yet SieveServe’s adaptive routing achieves it on all 16 pairs. On dense-information tasks specifically, SieveServe further retains $\geq 97.8\%$, confirming that the dual-guard mechanism provides additional safety margin where accuracy sensitivity is highest.

C QCR and DIG Configuration

Table 3 lists the hyperparameters for both guards. QCR uses a fine-tuned ModernBERT binary classifier (~ 400 MB) with confidence threshold $\gamma = 0.8$; queries with confidence below γ default to complex, ensuring conservative routing under uncertainty. DIG uses a single-scalar variance gate with threshold $\tau = 0.55$, calibrated on a held-out set of DocVQA, ChartQA, and InfoVQA images. Notably, DIG adopts a **conservative detection** strategy: the threshold is tuned so that some visually simple images (e.g., natural photos with uniform backgrounds) are also flagged as dense-info and routed to the bypass path. This design errs on the side of protection—an occasional missed speedup is preferable to an SLO violation—and explains why DIG Bypass% on pruning-robust benchmarks is non-zero (e.g., 10.2% on MME, 14.5% on MMBench) despite those images being safe to prune.

Table 4 reports per-benchmark routing statistics on Qwen2.5-VL, showing DIG bypass rate (fraction of images classified as dense-info) and QCR simple rate (fraction of queries routed to the aggressive path).

D Additional Guard and Workload Validation

Corrected Cap/DIG ablation. Table 5 reports the full P50/P90 values for the component ablation

Benchmark	DIG Bypass%	QCR Simple%	Keep%
POPE	0.4	24.1	30.9
MME	10.2	—	37.0
GQA	2.2	—	32.7
MMBench	14.5	—	32.1
DocVQA	94.0	—	98.8
InfoVQA	98.9	—	97.6
ChartQA	100.0	1.1	100.0
TextVQA	20.4	20.4	47.5

Table 4: Per-benchmark routing statistics (Qwen2.5-VL). DIG Bypass%: fraction of images flagged as dense-info. QCR Simple%: fraction of queries classified as simple. —: not separately measured.

Group	TTFT P50/P90	E2E P50/P90	OOM fallback
Vanilla	593.2 / 3489.6	2115.4 / 6217.7	0
Fixed HoloV	467.3 / 2986.2	1886.2 / 4810.3	62
+Cap (4K, no DIG)	394.3 / 1394.8	1747.0 / 2750.0	0
+DIG	443.3 / 1388.7	1792.2 / 2967.0	0

Table 5: Corrected component ablation on WildVision at QPS=2. Latencies are in milliseconds.

Method	Remove%	Overall	Pool A	Pool B
Vanilla vLLM	0.000	83.00	83.50	82.50
Fixed HoloV	66.700	73.50	66.63	80.37
Matched HoloV	39.592	74.38	68.75	80.00
SieveServe	39.592	81.19	81.87	80.50

Table 6: TraceA stationary mixed workload. Accuracy is reported in percent.

summarized in Figure 8.

TraceA and TraceB controls. Tables 6 and 7 report the full matched-budget trace results summarized in Figure 9. Pool A contains dense/high-risk requests; Pool B contains pruning-friendly requests. TraceB changes the Pool A:B mix from 1:5 to 1:1 to 5:1 while keeping the total arrival rate fixed.

DIG variance validation. Table 8 reports the full DIG validation summarized in Figure 10. Dense-route values are percentages.

Live-DIG versus replayed predictor. The main text summarizes the QPS=4 result; Table 9 reports the full load sweep. The replayed predictor uses the same route decisions as DIG, isolating the serving overhead of adding an image-level learned predictor to the online path.

E Additional Accuracy Controls

Aggressive fixed-ratio extension. Table 10 extends the fixed-ratio comparison on Qwen2.5-VL from the controlled 66.7% removal point to a more

Method	P1 Rem.	P2 Rem.	P3 Rem.	Global Rem.	P1 Acc.	P2 Acc.	P3 Acc.	Overall
Vanilla vLLM	0.00	0.00	0.00	0.00	87.36	89.61	86.54	87.62
Fixed HoloV	66.70	66.70	66.70	66.70	82.53	79.97	74.08	78.72
Matched HoloV	39.592	39.592	39.592	39.592	82.90	80.68	75.16	79.44
SieveServe	55.85	38.79	24.04	39.45	85.36	87.43	84.54	85.57

Table 7: TraceB composition-shift workload. Removal and accuracy are reported in percent.

Dataset	Qwen Var. Mean/P50	Qwen Dense-route%	Intern Var. Mean/P50	Intern Dense-route%
ChartQA	0.1806 / 0.1587	100.0	0.0493 / 0.0483	96.7
DocVQA	0.3013 / 0.2891	100.0	0.0507 / 0.0500	100.0
InfoVQA	0.4071 / 0.3906	88.9	0.0554 / 0.0559	92.6
TextVQA	0.6240 / 0.6250	16.7	0.0621 / 0.0620	50.0
GQA	0.7320 / 0.7402	0.0	0.0689 / 0.0693	10.0
MMBench	0.7772 / 0.7344	13.3	0.0694 / 0.0696	20.0
MME	0.7835 / 0.8633	6.7	0.0632 / 0.0618	53.3
POPE	0.7873 / 0.7930	0.0	0.0767 / 0.0757	0.0

Table 8: DIG variance validation on a balanced 192-image multi-benchmark set across two vision backbones.

QPS	Method	Success	TTFT Mean/P90	E2E Mean/P90
2	Live-DIG	160/160	1296.7 / 3144.0	2940.9 / 5416.4
2	Replay-Predictor	160/160	1420.2 / 3513.2	3045.3 / 5722.8
3	Live-DIG	160/160	1954.8 / 5117.7	4041.2 / 7647.1
3	Replay-Predictor	160/160	2374.6 / 5890.5	4716.6 / 8990.3
4	Live-DIG	160/160	3246.2 / 6719.5	6931.3 / 12207.4
4	Replay-Predictor	160/160	4338.0 / 8406.5	8977.7 / 16283.5

Table 9: Live-DIG versus replayed learned predictor on WildVision. Latencies are in milliseconds.

Method	Removal	LLaVA-Bench	VisIT-Bench	MM-Vet
Vanilla	—	92.99	7.111	64.817
SieveServe	adaptive	91.95	7.047	63.073
FastV	66.7%	92.27	6.926	53.404
HoloV	66.7%	84.82	6.774	54.220
DART	66.7%	82.77	6.807	49.450
DivPrune	66.7%	58.08	5.299	56.468
PiToMe	66.7%	59.30	5.379	38.899
FastV	88.9%	72.69	6.334	37.853
HoloV	88.9%	69.14	6.394	42.248
DART	88.9%	73.72	6.409	40.578
DivPrune	88.9%	44.27	4.381	42.294
PiToMe	88.9%	43.68	4.144	23.532

Table 11: Open-ended quality results. VisIT-Bench uses its native 1–10 scale; LLaVA-Bench and MM-Vet are reported as percentage-style scores.

aggressive 88.9% removal point. Across dense-information tasks, higher fixed removal consistently worsens accuracy, reinforcing that request-adaptive budget allocation is required for reliable pruning.

Task	FastV	DART	HoloV	DivPrune	PiToMe
DocVQA	91.5 → 60.8	77.9 → 42.6	67.8 → 44.7	16.9 → 12.4	16.7 → 11.1
TextVQA	82.0 → 74.6	72.2 → 52.2	71.3 → 49.4	26.6 → 22.6	26.9 → 21.9
ChartQA	76.7 → 50.4	63.0 → 33.5	54.0 → 31.4	12.7 → 9.1	11.9 → 8.3
InfoVQA	69.5 → 41.1	53.4 → 32.3	48.0 → 34.7	19.8 → 13.0	17.9 → 12.6

Table 10: Qwen2.5-VL fixed-ratio extension from 66.7% to 88.9% visual-token removal on dense-information tasks. Each cell reports score at 66.7% → score at 88.9%.

Open-ended quality. We further evaluate open-ended response quality using VisIT-Bench, LLaVA-Bench, and MM-Vet with a unified judge. Table 11 shows that SieveServe remains close to vanilla quality, while fixed-ratio baselines degrade substantially, especially at 88.9% removal.

F QCR Label Audit

Reviewer feedback raised whether the QCR training labels, which are obtained from an LLM-as-judge procedure, align with human intuition about query difficulty. We therefore performed a held-out audit on 200 queries that were not used for QCR classifier training. Five annotators independently labeled each query as requiring either a higher visual evidence budget (complex/high-resolution) or a lower visual evidence budget (simple/low-resolution); the human label is the majority vote. Table 12 compares the majority vote with the GLM-5 label used for weak supervision.

GLM-5 label / Human majority	Complex	Simple
Complex/high-resolution	122	2
Simple/low-resolution	6	70

Table 12: Agreement between GLM-5 QCR labels and five-annotator majority vote on 200 held-out queries. The majority vote agrees with GLM-5 on 192/200 queries (96.0%).

This audit is not intended to turn QCR into a semantic benchmark by itself. Its role is narrower: it checks whether the weak-supervision signal used for request routing is consistent with human judgments at the coarse granularity needed by SieveServe. Among the 20 examples with any minority disagreement, 12 were 3:2 in favor of the GLM-5 label, 6 were 3:2 against it, and 2 were 4:1 against it; no example had all five annotators opposing the GLM-5 label. This pattern suggests that most disagreements are boundary cases rather than systematic inversions of the simple/complex distinction.

G Multi-turn and Cache Reuse Scope

The present evaluation focuses on single-image, single-turn serving because this is the setting where visual-token pruning can be isolated cleanly from conversation-level cache policies. Nevertheless, SieveServe was designed around serving invariants that are directly relevant to multi-turn deployment. First, pruning is completed before LLM prefill and uses query-independent visual scoring, so the post-pruning visual sequence remains deterministic and compatible with prefix-cache reuse across repeated image contexts. Second, block alignment separates system-prompt and visual KV pages, allowing visual blocks to be accounted for, evicted, or rescued without corrupting text-prefix blocks. Third, the implementation already contains two cache-oriented mechanisms: an L0 prompt-agnostic embedding cache that reuses visual encoder outputs on repeated images, and an L2 host-DRAM rescue path for high-value visual KV blocks under GPU memory pressure.

What remains outside the current paper is a full conversation-level scheduler. In multi-turn or multi-image chat, the serving system must decide not only how many visual tokens to keep for one request, but also where to place a follow-up request so that existing image embeddings or visual KV blocks can be reused. A natural extension is therefore a cache-aware request scheduler that

routes later turns to GPUs or nodes holding the relevant visual cache, while coordinating offload and restore when the cache is remote. This is complementary to the current SieveServe contribution: the paper establishes that visual-token pruning can be made engine-compatible and request-adaptive; future work should combine it with session-aware cache placement and reuse policies for multi-turn workloads.

Such an extension also changes the evaluation target. For a single-turn workload, the key serving metrics are TTFT, TPOT, throughput, and whether post-pruning accuracy remains within the SLO. For a multi-turn workload, the scheduler must additionally track visual-cache hit rate, cross-turn cache migration cost, offload/restore latency, and whether pruning decisions remain stable when the same image is referenced by several follow-up questions. The current implementation exposes the relevant hooks—post-pruning token counts, visual-token block boundaries, L0 embedding reuse, and L2 rescue metadata—but we do not claim a complete multi-turn placement policy in this paper. We will release these integration points with the code so that future work can evaluate cache-aware scheduling policies on top of the same serving-compatible pruning substrate.