VPCache: Answer-Preserving Cache-View Selection for Repeated-Visual LVLM Inference

Anonymous authors
Paper under double-blind review

Abstract

Repeated-visual LVLM inference answers multiple questions about the same image, document, chart, or screenshot. Exact prefix reuse saves repeated encoding but retains the full visual representation, while compression can discard evidence needed by a later question. We propose VPCache, which selects reusable cache views according to paired answer sufficiency. Its Answer-Preserving Cache View Selector (AP-CVS) qualifies candidate views within question cells, prioritizes answer quality, and freezes the resulting routes before evaluation. A learned high-fidelity view, APVC-hifi, refines a deterministic anchor for reuse across questions. On 448 held-out requests from seven workloads with Qwen2.5-VL-7B-Instruct, VPCache adds two soft and two exact matches over the guarded base, with no paired soft losses against either the base or full-prefix inference. Static compression controls incur 25–72 paired soft losses against the base. Across all held-out requests, VPCache reduces injected visual tokens by 4.90% and selected-view memory slots by 2.46% on average, matching the base's representation budget while improving answer quality. These results support question-cell selection as a way to govern reusable visual compression.

1 Introduction

Large vision-language models (LVLMs) provide a common interface for visual perception, grounded dialogue, and multimodal reasoning (Alayrac et al., 2022; Li et al., 2023). Visual instruction tuning extends them to open-ended question answering (Liu et al., 2023). Recent high-resolution architectures also support document parsing, chart interpretation, and complex layouts (Bai et al., 2025). These capabilities carry a substantial context cost: the image becomes a visual prefix that is processed during prefill and retained during decoding. Prefix-aware serving systems reuse shared contexts and improve cache management (Kwon et al., 2023; Zheng et al., 2024), but exact reuse retains the full representation. This cost matters when many questions share a document, chart, product image, or screenshot.

Visual-token and KV-cache compression reduce this cost. LLaVA-PruMerge selects important visual tokens and merges spatially redundant ones before multimodal decoding (Shang et al., 2025). SparseVLM removes tokens using text-guided relevance without additional training (Zhang et al., 2025). TokenPacker compresses visual features through an efficient projector (Li et al., 2025), and FasterVLM uses visual-encoder attention for training-free pruning (Zhang et al., 2024). At the cache level, KIVI uses low-bit KV quantization (Liu et al., 2024), and SnapKV retains a compact set of important positions (Li et al., 2024). VL-Cache allocates cache capacity across modalities and layers (Tu et al., 2025). These methods expose substantial visual redundancy and offer useful candidate representations. Reusing such representations across questions raises a further decision: which requests can use each compressed view without losing answer-relevant evidence?

Visual redundancy depends on the question. TextVQA requires scene text that may occupy a small image region (Singh et al., 2019). DocVQA and InfographicVQA also require dense text, layout, and spatial relations (Mathew et al., 2021; 2022); ChartQA combines visual lookup with numerical and logical reasoning (Masry et al., 2022). A view sufficient for a coarse semantic question may lose evidence needed for OCR, counting, chart reading, or fine-grained attributes. Selective prediction

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

screens unreliable decisions (Geifman & El-Yaniv, 2017), confidence calibration supports decision thresholds (Guo et al., 2017), and cost-aware routing selects among model executions (Chen et al., 2024). Applying these ideas to a single LVLM requires determining whether a compressed view preserves correctness relative to a reference execution. The resulting problem is to select an answer-sufficient view for the current question and determine when to fall back.

We propose VPCache, an answer-preserving cache-view selection framework for repeated-visual LVLM inference. Its view bank contains exact reuse, deterministic memories, static token or KV compression, and a learned Answer-Preserving Cache View Compressor (APVC). APVC builds a high-fidelity memory around a deterministic anchor to preserve evidence for later questions. An Answer-Preserving Cache View Selector (AP-CVS) evaluates candidates on a development split and freezes a quality-first route for each question cell, using representation cost to break quality ties. Preservation is measured by paired soft-match regressions against FacetGuard-base, a deterministic guarded policy; exact-match quality is tracked separately. Each inference request follows one fixed route, with FacetGuard-base as fallback. We evaluate VPCache with Qwen2.5-VL-7B-Instruct on seven repeated-image QA workloads in a shared pipeline. On 448 held-out requests, static compression controls introduce 25–72 paired soft losses. AP-CVS selects APVC-hifi for binary visual questions and a KIVI-style view for infographic counting. The frozen selector gains two soft and two exact matches over the guarded base, with zero paired soft losses against both the base and full-prefix inference.

Contributions. Our contributions are:

- We formulate repeated-visual LVLM inference as *answer-preserving cache-view selection*, unifying exact, deterministic, static-compressed, and learned cache representations within a common view bank.
- We propose AP-CVS, which qualifies candidate views by paired correctness, freezes a quality-first route for each question cell, and retains the guarded base for unsupported cells.
- We introduce APVC, a learned high-fidelity cache view that retains reusable answer-relevant evidence around a deterministic anchor.
- We evaluate held-out answer changes against both guarded and full references, and account for representation reductions over every request. The resulting comparison exposes the trade-off between compression coverage and paired answer regressions.

2 Related Work

Efficient LVLM inference and visual token compression. Modern LVLMs add a visual prefix to the long context processed by a language model. Its size grows with image resolution, multiple images, and document-like scenes. Token compression reduces this prefix before or during decoding. PruMerge selects important tokens and merges redundant ones in LLaVA-style models (Shang et al., 2025); SparseVLM uses relevance from visual-text attention (Zhang et al., 2025); Task-Related Token Compression estimates instruction-related relevance with explainability tools (Lei et al., 2026); and VisionSelector learns selection through differentiable Top-K training (Zhu et al., 2025). LightKV compresses visual KV states during prefill through prompt-aware cross-modal message passing (Chen et al., 2026). These methods exploit visual redundancy. VPCache addresses the deployment decision that follows: which compressed view can serve a given question cell, and when should the system fall back?

Reusable visual representations. Compress & Cache learns compact visual summaries in an upfront indexing stage and reuses them for later generation and retrieval (Bulat et al., 2026). Its two-pass training combines autoregressive and contrastive objectives. This shares VPCache's motivation of amortizing visual compression across later queries. VPCache's central contribution is the selection layer: candidate views are qualified by paired answer changes within question cells and deployed through a frozen policy. APVC-hifi is one candidate in this bank, using an anchored feature-space objective.

KV-cache compression and LLM serving caches. KV-cache optimization reduces the cost of long-context LLM serving. PagedAttention improves memory management (Kwon et al., 2023),

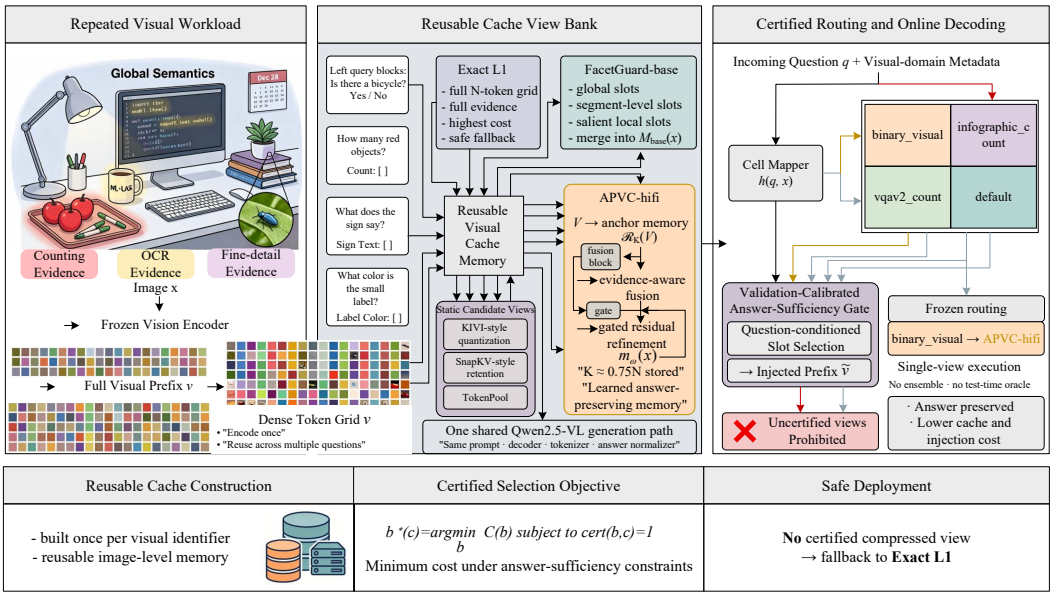


Figure 1: VPCache separates reusable view construction from question-time routing. AP-CVS qualifies candidates against FacetGuard-base and ranks them by soft match, exact match, and representation cost. FacetGuard-base is the selector fallback and uses Exact L1 internally for requests outside its compression policy.

and SGLang combines structured execution with prefix reuse (Zheng et al., 2024). KIVI applies asymmetric low-bit quantization (Liu et al., 2024); SnapKV retains clustered cache positions selected from a prompt observation window (Li et al., 2024). These techniques complement our setting. KIVI-style quantization, SnapKV-style retention, and token pooling form candidate views in our bank. VPCache adds the answer-sufficiency layer that determines whether each view can serve a calibrated question cell. VL-Cache uses modality- and layer-aware sparsity to compress VLM KV states (Tu et al., 2025). Our experiments use visual-prefix controls implemented in the shared runner; their role is to isolate the effect of selecting among views with different correctness profiles.

3 Method

VPCache builds reusable visual representations and selects one route per question cell. AP-CVS qualifies candidates against a deterministic reference, then freezes the policy before evaluation. We use *certified* to denote passing the empirical criterion in Equation (11).

3.1 Answer-Preserving Cache-View Formulation

Given image x, the frozen vision encoder produces

$$V = f_\theta^{\mathrm{vis}}(x) = [v_1, \ldots, v_N] \in \mathbb{R}^{N \times d}, \tag{1}$$

where v_i is a visual token, N is the token count, and d is the hidden dimension. The view bank is

$$\mathcal{B} = \{b_{\mathrm{exact}}, b_{\mathrm{base}}, b_1, \ldots, b_m, b_{\mathrm{APVC}}\}, \tag{2}$$

where Exact L1 retains the full prefix, FacetGuard-base is a deterministic guarded policy, $b_1, \ldots, b_m$ are static compression views, and b_{APVC} is the learned APVC-hifi view. Each view defines an image-level memory constructor ϕ_b and a question-time injection operator g_b:

$$M_b(x) = \phi_b(V), \qquad \widetilde{V}_b(x, q) = g_b(M_b(x), q). \tag{3}$$

Exact L1 has $M_b(x) = \widetilde{V}_b(x, q) = V$. FacetGuard-base uses a multi-facet memory for eligible requests and Exact L1 otherwise. Its answers can therefore differ from full-prefix inference.

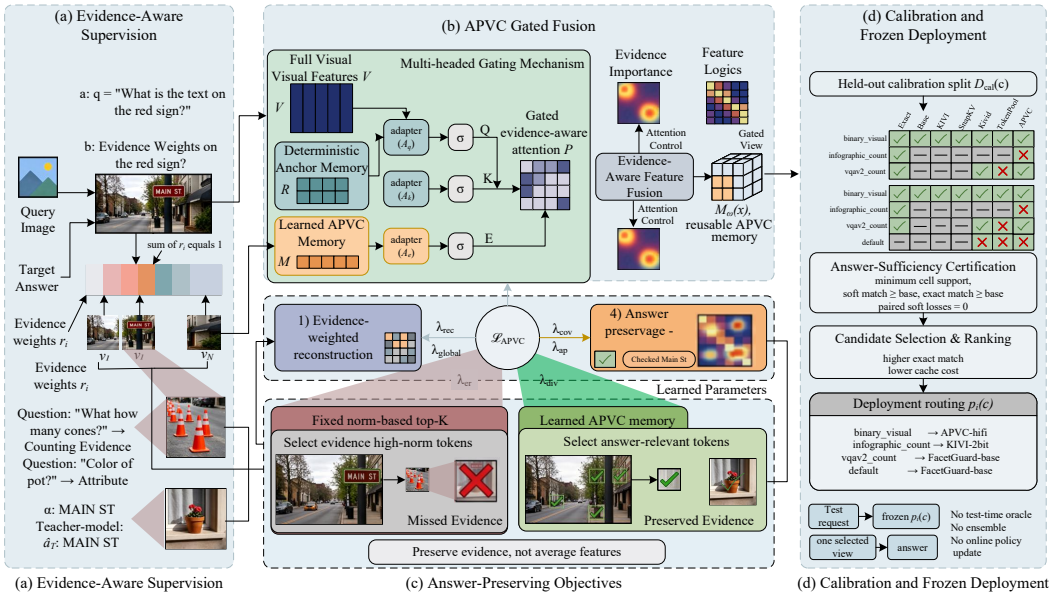


Figure 2: APVC-hifi refines a deterministic anchor through gated attention over the full visual prefix. Answer-conditioned feature-space objectives supervise training. At inference, memory construction depends on the image, and question-time slot selection uses the incoming question.

A fixed mapper assigns a request to a question cell:

$$c = h(q, x) \in \mathcal{C}, \tag{4}$$

where h uses question form and visual-domain metadata. Let $\mathcal{B}_{\text{cand}}$ contain the static and learned candidates. The admissible set is

$$\mathcal{B}_{\text{adm}}(c) = \{b_{\text{base}}\} \cup \{b \in \mathcal{B}_{\text{cand}} : \text{cert}(b, c) = 1\}. \tag{5}$$

Here $\text{cert}(b, c)$ tests paired soft-match preservation and aggregate soft/exact non-inferiority against the base. AP-CVS ranks this set by answer quality, using representation cost to break ties (Equation (12)). Cell-level qualification shares development evidence across requests, while deployment executes one assigned route.

3.2 Answer-Preserving Cache View Compressor

APVC refines a deterministic anchor into an image-level memory reusable across questions (Figure 2). For a training tuple $(x, q, a, \widehat{a}_F)$, normalized evidence weights are

$$r_i = \frac{\exp\big(\psi(v_i; q, a, \widehat{a}_F)/t_r\big)}{\sum_{j=1}^{N} \exp\big(\psi(v_j; q, a, \widehat{a}_F)/t_r\big)}, \tag{6}$$

where a is the target answer, $\widehat{a}_F$ is the recorded reference answer, and $t_r > 0$ is a temperature. The fixed scorer ψ combines a deterministic projection of the question and answers with token norm and local feature variation. It supplies training weights with $\sum_i r_i = 1$ and is distinct from the learned evidence head below. Section A.1 specifies the scorer and training split.

For memory budget K, the anchor is

$$R = \mathcal{R}_K(V) \in \mathbb{R}^{K \times d}, \tag{7}$$

where $\mathcal{R}_K$ concatenates a global mean, pooled sequence segments, and high-norm local tokens. APVC-hifi stores approximately $K = 0.75N$ slots and selects approximately $0.5N$ for injection on

eligible requests. With $R_0 = \mathrm{LN}(R)$ and $V_0 = \mathrm{LN}(V)$, gated fusion produces

$$\begin{aligned} Q &= \mathrm{unit}(R_0 + \sigma(\gamma_q) A_q(R_0)), \\ K_V &= \mathrm{unit}(V_0 + \sigma(\gamma_k) A_k(V_0)), \\ E &= A_e(V_0), \\ P &= \mathrm{softmax}_N\big(QK_V^\top/\tau + \sigma(\gamma_e)\mathbf{1}_K E^\top\big), \\ M_\omega(x) &= R + \sigma(\gamma_f)\,[PV - R + A_o(PV)]. \end{aligned} \tag{8}$$

LN denotes learned layer normalization, and unit normalizes each row to unit length. The query and key matrices have shapes $Q \in \mathbb{R}^{K\times d}$ and $K_V \in \mathbb{R}^{N\times d}$. The evidence head gives $E \in \mathbb{R}^{N\times 1}$, broadcast over slots by $\mathbf{1}_K E^\top$. Softmax acts over the N source tokens, giving $P \in \mathbb{R}^{K\times N}$. The A's are lightweight adapters, the γ's are learned scalar gates, σ is sigmoid, and $\tau = 0.07$. Their parameters ω produce the reusable memory $M_\omega(x) \in \mathbb{R}^{K\times d}$.

Training combines feature reconstruction, evidence retention, and regularization:

$$\begin{aligned} \mathcal{L}_{\mathrm{APVC}} = s\big[&0.55\mathcal{L}_{\mathrm{rec}} + 0.20\mathcal{L}_{\mathrm{global}} + (0.35 + 0.15\rho)\mathcal{L}_{\mathrm{cov}} \\ &+ (0.25 + 0.20\rho)\mathcal{L}_{\mathrm{ans}} + 0.20\mathcal{L}_{\mathrm{moment}} + 0.15\rho\mathcal{L}_{\mathrm{hard}} \\ &+ 0.10\mathcal{L}_{\mathrm{margin}} + 0.08\mathcal{L}_{\mathrm{anchor}} + 0.01\mathcal{L}_{\mathrm{div}}\big]. \end{aligned} \tag{9}$$

The reconstruction terms preserve evidence-weighted features and the full prefix. Coverage, answer-anchor, moment, hard-evidence, and margin terms retain selected feature directions; anchor and diversity terms regularize the memory. These are feature-space objectives. The sample weight s and risk score ρ are fixed functions of question type and reference-answer quality. Section A.1 gives their definitions. Only the compressor is optimized; the LVLM remains frozen.

3.3 Answer-Sufficiency Calibration

For cell c, let $\mathcal{D}_{\mathrm{cal}}(c)$ be the development examples used for route selection. For candidate b, define

$$\begin{aligned} S_b(c) &= \sum_{(x,q,a)\in\mathcal{D}_{\mathrm{cal}}(c)} \delta_{\mathrm{soft}}(\widehat{a}_b, a), \\ E_b(c) &= \sum_{(x,q,a)\in\mathcal{D}_{\mathrm{cal}}(c)} \delta_{\mathrm{exact}}(\widehat{a}_b, a), \\ L_b(c) &= \sum_{(x,q,a)\in\mathcal{D}_{\mathrm{cal}}(c)} \mathbb{I}[\delta_{\mathrm{soft}}(\widehat{a}_{\mathrm{base}}, a) = 1,\ \delta_{\mathrm{soft}}(\widehat{a}_b, a) = 0]. \end{aligned} \tag{10}$$

Here $\widehat{a}_b$ is the answer from view b; δ_{soft} and δ_{exact} are the runner's binary answer-match indicators. S_b and E_b count soft and exact matches, while L_b counts base-correct requests that become incorrect under the soft criterion.

A candidate is certified when

$$\mathrm{cert}(b,c) = \mathbb{I}\begin{bmatrix} n_c \ge n_{\min}, & S_b(c) \ge S_{\mathrm{base}}(c), \\ E_b(c) \ge E_{\mathrm{base}}(c), & L_b(c) = 0 \end{bmatrix}, \tag{11}$$

where $n_c = |\mathcal{D}_{\mathrm{cal}}(c)|$, $n_{\min} = 8$ is the minimum support, and the reference counts come from FacetGuard-base. AP-CVS selects

$$\pi(c) = \underset{b\in\mathcal{B}_{\mathrm{adm}}(c)}{\arg\max}\big(S_b(c), E_b(c), -C_c(b)\big), \tag{12}$$

with lexicographic comparison: soft match first, exact match second, and lower representation cost third. In the reported implementation, $C_c(b)$ is the mean retained-memory token ratio on the cell's development requests. A complete tie retains the base when it is tied for best; if no candidate qualifies, $\pi(c) = b_{\mathrm{base}}$. The guard constrains paired soft losses and aggregate exact count. Evaluation reports paired exact wins and losses separately.

3.4 Frozen Deployment

For a request (x, q), the frozen policy injects

$$\widetilde{V}(x, q) = g_{\pi(h(q,x))}\big(M_{\pi(h(q,x))}(x), q\big)\,. \tag{13}$$

The decoder generates one answer with greedy next-token selection:

$$\widehat{a}_t = \arg\max_z p_\theta^{\mathrm{dec}}(z \mid \widehat{a}_{<t}, q, \widetilde{V}(x, q)). \tag{14}$$

Here t is the generation step, z ranges over the vocabulary, and $\widehat{a}_{<t}$ is the generated prefix. APVC-hifi serves binary visual questions, the KIVI-style view serves infographic counting, and other cells retain FacetGuard-base. Both guarded views retain their internal Exact L1 fallback. View assignment and actual prefix compression are therefore reported separately.

4 Experiments

We test whether selecting reusable views preserves paired answer correctness and how much representation reduction remains across the full workload. The primary comparison uses 448 held-out requests; the combined 896-request matrix provides a development diagnostic.

4.1 Experimental Setup

Model and workloads. All routes use Qwen2.5-VL-7B-Instruct (Bai et al., 2025), greedy decoding, a maximum of 32 generated tokens, and an image budget of 262,144 pixels. The seven workloads are ABO product QA (Collins et al., 2022), AI2D (Kembhavi et al., 2016), ChartQA (Masry et al., 2022), DocVQA (Mathew et al., 2021), InfographicVQA (Mathew et al., 2022), TextVQA (Singh et al., 2019), and VQAv2 (Goyal et al., 2017). Each contributes 128 requests, totaling 896 requests over 304 visual inputs (2.95 questions per input on average). Dataloaders, prompts, decoding limits, and the answer matcher are shared across routes.

Training, route selection, and evaluation. The first 64 requests per workload form the development split. APVC-hifi trains on these 448 requests over 155 visual inputs for two epochs with AdamW at learning rate 5×10^{-4}, using target answers and recorded FacetGuard-base predictions. The same development split supplies AP-CVS route selection; its image–question deduplication leaves 447 unique pairs. The remaining 448 requests are held out from parameter training and route selection. They cover 152 visual inputs, three of which also occur in the development split. This is a request-level split. The checkpoint, scoring, and cell rules are detailed in Sections A.1 and A.2.

Compared routes. Fresh inference computes the full prefix without reuse; Exact L1 reuses that prefix. Their predictions agree on all 896 requests. Static controls in the shared runner implement KIVI-style 2-bit fake quantization of visual features, SnapKV-style window retention, and TokenPool. We label the first two as proxies for their respective compression principles (Liu et al., 2024; Li et al., 2024). FacetGuard-base combines deterministic multi-facet memory with Exact L1 fallback. VPCache adds the frozen AP-CVS policy and APVC-hifi. Each deployed request decodes one selected route.

Metrics. Exact match is equality after answer normalization. Soft match accepts exact match, substring containment in either direction, or a numeric match within $\max(10^{-4}, 10^{-3}|a_{\mathrm{num}}|)$ of a reference number. These are shared runner metrics, applied uniformly across workloads (Section A.2). Paired wins/losses count changes on the same request and are reported for both metrics. Representation reductions average $1 - N_{\mathrm{injected}}/N_{\mathrm{full}}$ and $1 - K_{\mathrm{selected}}/N_{\mathrm{full}}$ over *all* requests, with zero reduction on exact paths. They describe the selected representation; total resident cache bytes also depend on which views coexist for each image.

4.2 Held-Out Results

Paired preservation and the full-prefix reference. Table 1 shows that the frozen VPCache policy gains two soft and two exact matches over FacetGuard-base, with zero paired soft loss. The static

Route	Soft	Exact	Δ soft/exact	Soft W/L	Exact W/L
Fresh / Exact L1	283	176	$-2/+6$	0/2	6/0
KIVI-style proxy	270	169	$-15/-1$	10/25	15/16
SnapKV-style proxy	223	114	$-62/-56$	10/72	11/67
TokenPool	241	121	$-44/-49$	11/55	11/60
FacetGuard-base	285	170	–	–	–
VPCache	**287**	172	$+2/+2$	**2/0**	3/1

Table 1: Primary comparison on 448 held-out requests (64 per workload). Deltas and paired wins/losses use FacetGuard-base as reference. Soft and exact regressions are counted separately.

controls introduce 25, 72, and 55 soft losses, respectively. Against Fresh or Exact L1, FacetGuard-base has soft W/L of 2/0 and exact W/L of 0/6; VPCache has soft W/L of 4/0 and exact W/L of 2/6. Thus, the observed soft-preservation result holds against the full visual route as well as the guarded base. Exact preservation is a separate outcome: VPCache improves the base's aggregate exact count while retaining six exact regressions against full-prefix inference.

Cell	Requests	Selected route	Δ soft/exact	Soft W/L
Binary visual	30	APVC-hifi	$0/+1$	0/0
Infographic count	21	KIVI-style	$+2/+1$	2/0
VQAv2 count	9	Base	$0/0$	0/0
Default	388	Base	$0/0$	0/0

Table 2: Frozen routing on the 448 held-out requests. The two non-base assignments cover 51 requests (11.4%). A guarded route can still execute Exact L1 internally.

Coverage and workload-level representation cost. AP-CVS assigns 30 held-out requests to APVC-hifi, 21 to the KIVI-style view, and 397 to FacetGuard-base (Table 2). Across these routes, 44 requests actually inject a shorter visual prefix. Their mean token and memory-slot reductions are 49.9% and 25.0%; averaging over all 448 requests gives 4.90% and 2.46%. FacetGuard-base has the same reductions, so the selector improves answer quality at the base's representation budget. The KIVI-style proxy retains the visual-token count and simulates quantization in floating-point tensors. The measured reductions therefore come from the guarded memory paths. Controlled latency and total resident cache bytes require matched cache states and accounting for coexisting views.

Workload dependence. Table 3 localizes the changes. VPCache retains the base's counts on five workloads, adds two soft and one exact match on InfographicVQA, and adds one exact match on VQAv2. The static controls lose more answers on text- and layout-intensive workloads even when their aggregate counts improve on a different workload. This heterogeneity motivates qualifying views by cell and paired correctness.

Workload	Fresh	KIVI-style	SnapKV-style	TokenPool	Base	VPCache
ABO-MVR	9/1	8/1	8/0	8/0	9/1	9/1
AI2D	50/0	50/0	45/0	47/0	50/0	50/0
ChartQA	47/32	42/31	44/31	37/24	47/32	47/32
DocVQA	51/45	47/41	30/21	38/30	51/44	51/44
InfographicVQA	37/31	35/31	21/17	28/22	37/31	**39/32**
TextVQA	44/31	43/30	26/16	36/18	44/30	44/30
VQAv2	45/36	45/35	49/29	47/27	47/32	**47/33**

Table 3: Held-out soft/exact counts, with 64 requests per workload. KIVI-style and SnapKV-style columns are in-runner proxies.

4.3 Compressor and Development Diagnostics

APVC compressor design. Table 4 compares APVC configurations over the combined 896-request matrix. The full configuration preserves the base soft count and adds three exact matches, including

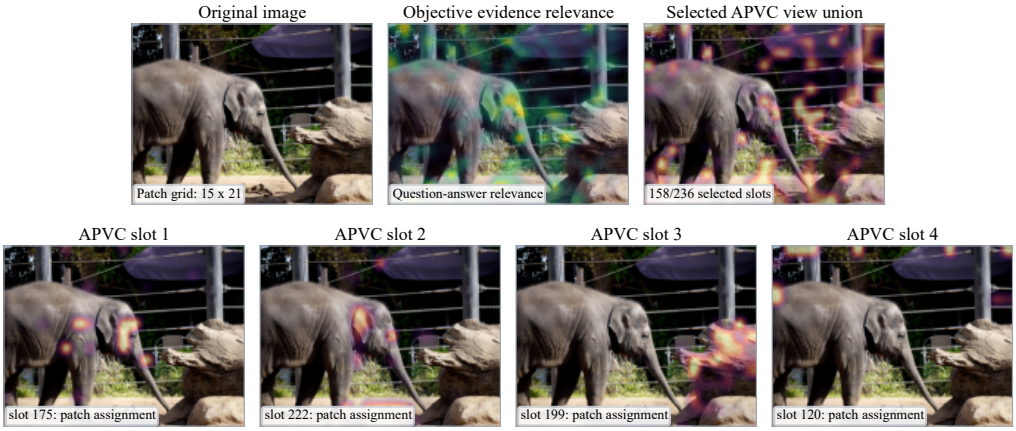


Figure 3: APVC memory on a VQAv2 example. Top: original image, fixed training-evidence weights, and the union of selected slots. Bottom: four slot-to-patch assignment maps. The 158 selected slots are drawn from a 236-slot memory built from 315 source tokens.

one on the held-out split. The aggressive 0.50-memory variants incur six or seven paired soft losses; training at a 0.50 budget and deploying at 0.75 leaves one. These results favor training at the deployed memory budget and using APVC-hifi as a selectively admitted view.

Configuration	Train K/N	Eval. K/N	Soft/Exact	Δ soft/exact	Soft W/L
Deterministic base	–	0.75	579/344	–	–
APVC-v1	0.50	0.50	572/342	$-7/-2$	0/7
APVC-v2	0.50	0.50	573/343	$-6/-1$	0/6
APVC-v2	0.50	0.75	578/347	$-1/+3$	0/1
APVC-hifi	0.75	0.75	**579/347**	$0/+3$	**0/0**

Table 4: APVC development diagnostic on all 896 requests, including training and route-selection data. W/L consistently denotes paired *soft* changes against the base. K/N is the stored-slot budget on the compressed path. APVC-v1 is an earlier objective configuration.

Spatial structure of the learned memory. Figure 3 visualizes the APVC memory for a VQAv2 image with 315 visual tokens. The compressor stores 236 slots and selects 158 for the incoming question. Individual slots assign weight to distinct regions: two emphasize the elephant's head and body, while others cover the adjacent rocks and background. Their union retains both the subject and surrounding context. The maps show the compressor's slot-to-patch assignments; the paired answer comparisons measure whether that representation remains sufficient for decoding.

Combined matrix and resolution check. The frozen policy adds $+2/+3$ soft/exact matches on development requests and $+2/+2$ on held-out requests. The combined $+4/+5$ result and the corresponding per-workload diagnostic appear in Section A.3. At a lower 131,072-pixel budget, APVC-hifi gains $+1/+1$ on the held-out split with zero paired soft loss, while its development split contains one loss (Table 5). This check illustrates why a change in image budget should be accompanied by renewed route qualification.

Split	n	Base	APVC-hifi	Soft W/L	Exact W/L
Development	448	260/140	261/141	2/1	2/1
Held-out	448	256/141	257/142	1/0	1/0

Table 5: APVC-hifi at a 131,072-pixel budget. Counts are soft/exact; paired W/L compares APVC-hifi with the base on the same requests.

4.4 Selector and Compressor Contributions

Held-out component comparison. Table 6 separates AP-CVS and APVC on the same 448 held-out requests. The static selector removes APVC from the candidate bank, applies the same qualification and ranking rules on development data, and freezes the resulting policy. It selects KIVI-style for infographic counting and the base for all other cells. APVC-only uses the learned memory within the guarded execution path, without AP-CVS selection among views. The full VPCache policy combines both components.

Configuration	Soft	Exact	Soft W/L	Exact W/L
FacetGuard-base	285	170	–	–
AP-CVS, static bank	287	171	2/0	2/1
APVC-only	285	171	0/0	1/0
VPCache	**287**	**172**	**2/0**	3/1

Table 6: Component comparison on 448 held-out requests. Paired W/L uses FacetGuard-base. All four configurations have the same mean injected-token and selected-memory-slot reductions, 4.90% and 2.46%.

The selector with a static bank contributes two soft matches and a net gain of one exact match. APVC contributes one additional exact match while preserving all paired soft outcomes. Their combination retains both gains at the same representation budget. This decomposition assigns the soft improvement to selection among complementary views and the remaining exact improvement to the learned representation. The paired exact column also shows that the selector's net exact gain includes both wins and a loss.

Development evidence for route selection. Table 7 exposes the development counts behind the frozen policy. In the binary cell, KIVI-style has one more soft match than the base but one fewer exact match, so it fails the aggregate exact non-inferiority condition. APVC preserves the soft count and adds two exact matches with zero paired soft loss, making it admissible and preferred. For infographic counting, KIVI-style adds two soft matches and one exact match with zero paired soft loss, and wins the quality-first comparison.

Cell	n	Base	KIVI-style	APVC	Selected
Binary visual	29	25/17	26/16	25/19	APVC
Infographic count	12	8/6	10/7	8/6	KIVI-style
Default	402	258/148	232/134	258/148	Base
VQAv2 count	4	2/2	2/2	2/2	Base

Table 7: Development soft/exact counts for the base and the two views used by VPCache. Qualification also requires zero paired soft loss and at least eight development pairs.

The default cell retains the base because APVC ties it on quality and representation cost. The VQAv2-count cell has only four development pairs, so the minimum-support rule retains the base. These decisions yield two non-base routes, applied to 51 held-out requests; the remaining requests retain the base policy. Qualitative examples of the resulting answers appear in Section A.7.

5 Conclusion

VPCache treats repeated-visual inference as selection among reusable cache views. AP-CVS qualifies views by paired soft-match preservation and aggregate exact-match quality, then freezes a quality-first policy with a guarded fallback. APVC-hifi supplies a learned representation at the deterministic base's memory budget. On 448 held-out requests, the combined policy improves the base by two soft and two exact matches with no paired soft loss; the same soft-loss result holds against full-prefix inference. The observed 4.90% mean reduction in injected visual tokens reflects the coverage of the guarded execution paths. Extending that coverage through stronger candidate views, additional backbones, and controlled serving measurements is the next step toward larger practical gains.

AI USE STATEMENT

Generative AI tools were used solely to assist with and polish the writing of this manuscript. The authors reviewed and revised all AI-assisted content and take full responsibility for the final manuscript.

REPRODUCIBILITY STATEMENT

The paper specifies the Qwen2.5-VL-7B-Instruct inference setup, seven repeated-visual workloads, 896 paired examples, exact and compressed cache views, APVC, AP-CVS calibration, preservation criteria, and the frozen routing policy. The appendix provides the final selector, component runs, workload details, ablations, and memory diagnostics. This repository contains the manuscript source, figures, tables, and references; it does not include the complete inference harness, cache contents, model checkpoints, or all question and answer records. Reproducing the results requires those artifacts, the specified model version, public or licensed workloads, and the evaluation settings described in the paper.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L. Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Ming-Hsuan Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025.

Adrian Bulat, Yassine Ouali, and Georgios Tzimiropoulos. Compress & cache: Vision token compression for efficient generation and retrieval. *Advances in Neural Information Processing Systems*, 38:31943–31968, 2026.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *Trans. Mach. Learn. Res.*, 2024, 2024.

Xihao Chen, Yangyang Guo, and Roger Zimmermann. Make your LVLM KV cache more lightweight. *Trans. Mach. Learn. Res.*, 2026, 2026.

Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, et al. Abo: Dataset and benchmarks for real-world 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 21126–21136, 2022.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. *Advances in neural information processing systems*, 30, 2017.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 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.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European conference on computer vision*, pp. 235–251. Springer, 2016.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Lei Lei, Jie Gu, Xiaokang Ma, Chu Tang, Jingmin Chen, and Tong Xu. Task-related token compression in multimodal large language models from an explainability perspective. In *The Fourteenth International Conference on Learning Representations*, 2026.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pp. 19730–19742. PMLR, 2023.

Wentong Li, Yuqian Yuan, Jian Liu, Dongqi Tang, Song Wang, Jie Qin, Jianke Zhu, and Lei Zhang. Tokenpacker: Efficient visual projector for multimodal llm. *International Journal of Computer Vision*, 133(10):6794–6812, 2025.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen (Henry) Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *ICML*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32332–32344. PMLR / OpenReview.net, 2024.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq R. Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *ACL (Findings)*, volume ACL 2022 of *Findings of ACL*, pp. 2263–2279. Association for Computational Linguistics, 2022.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 2200–2209, 2021.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and CV Jawahar. Infographicvqa. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1697–1706, 2022.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *ICCV*, pp. 22857–22867. IEEE, 2025.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8317–8326, 2019.

Dezhan Tu, Danylo Vashchilenko, Yuzhe Lu, and Panpan Xu. Vl-cache: Sparsity and modality-aware KV cache compression for vision-language model inference acceleration. In *ICLR*. OpenReview.net, 2025.

Qizhe Zhang, Aosong Cheng, Ming Lu, Zhiyong Zhuo, Minqi Wang, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. [CLS] attention is all you need for training-free visual token pruning: Make VLM inference faster. *CoRR*, abs/2412.01818, 2024.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis A. Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. Sparsevlm: Visual token sparsification for efficient vision-language model inference. In *ICML*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

Jiaying Zhu, Yurui Zhu, Xin Lu, Wenrui Yan, Dong Liu, Kunlin Liu, Xueyang Fu, and Zheng-Jun Zha. Visionselector: End-to-end learnable visual token compression for efficient multimodal llms. *CoRR*, abs/2510.16598, 2025.

A Appendix

A.1 Training and Data Provenance

APVC-hifi uses the `apvc_v2_l64_m075_e2_20260527` checkpoint. Its archived metadata records 448 training requests, 155 visual inputs, two epochs, AdamW learning rate 5×10^{-4}, a 0.75 memory ratio, a 0.50 injection ratio, and a 262,144-pixel image budget. The training loader takes the first 64 records from each workload. It looks up the target answer and the same-condition FacetGuard-base prediction for each training request; all 448 requests are retained, including reference-incorrect examples. The vision encoder and language decoder remain frozen. The learned evidence head, query/key/output adapters, layer norms, and scalar gates are optimized jointly through the feature-space objective.

The development requests also supply the AP-CVS qualification counts. One DocVQA image–question pair occurs twice at indices 16 and 17; the route selection script keys records by image and question and therefore uses 447 unique development pairs. Request-level tables preserve both occurrences and pair methods by workload and request index, yielding 896 total records. The held-out half has 448 unique image–question pairs. It spans 152 images; seven requests concern three images also present in the development half. This split measures transfer to held-out questions in a repeated-visual workload. A split by image would measure transfer to unseen visual inputs. The archived combined L128 diagnostics also informed development of the memory-budget and objective configurations; the held-out designation here refers to parameter fitting and cell-route selection.

Fixed training weights. The scorer ψ in Equation (6) uses a deterministic text projection: concatenate the question, question class, target answer, and reference answer when it differs from the target; map the UTF-8 string to a d-dimensional vector using repeated SHA-256 digests, with bytes rescaled to $[-1, 1]$. Its cosine score with each visual feature is combined with the normalized feature norm and a detail score. The detail score sums distance from the global feature mean and distance from the preceding token, then normalizes by its maximum. With question-class risk u, the norm and detail coefficients are $0.05 + 0.08u$ and $0.10 + 0.20u$, and $t_r = \max(0.12, 0.25 - 0.08u)$. The implementation scales the normalized weights by N and renormalizes within weighted losses. The scorer has no trainable parameters; it is a heuristic weighting function rather than a learned text–vision alignment model.

The class risk u is 1 for OCR, chart/table, numeric, and counting questions, 0.75 for binary/spatial questions, 0.35 for semantic questions, and 0.50 otherwise. The loss risk ρ raises u to at least 0.85 for numeric text, 0.70 for binary/spatial terms, and 0.65 for target answers of at most three characters. The sample weight s is 1.25 for reference-exact examples or 1 for reference-soft-only examples, plus 0.35 for high-risk classes or 0.15 for binary/spatial classes; reference-incorrect examples have weight 0.25.

Feature-space losses. $\mathcal{L}_{\mathrm{rec}}$ reconstructs full features from the injected slots by cosine-attention weighting and uses evidence-weighted mean squared error. $\mathcal{L}_{\mathrm{global}}$ applies unweighted reconstruction error using all stored slots. $\mathcal{L}_{\mathrm{cov}}$ weights each source token's distance to its most similar injected slot. $\mathcal{L}_{\mathrm{ans}}$ applies this nearest-slot penalty to the top evidence-weighted tokens, using a budget of one quarter of the injected slots. $\mathcal{L}_{\mathrm{hard}}$ uses a larger, risk-dependent evidence subset. $\mathcal{L}_{\mathrm{moment}}$ matches weighted feature means and standard deviations. $\mathcal{L}_{\mathrm{margin}}$ encourages an injected slot to retain the source's score along the fixed text projection through a softplus margin. Finally, $\mathcal{L}_{\mathrm{anchor}}$ penalizes squared deviation from the deterministic memory and $\mathcal{L}_{\mathrm{div}}$ penalizes redundant slot directions. Target answers enter these training weights and feature objectives; inference memory construction uses the image alone.

A.2 Answer Metrics and Question Cells

The shared answer normalizer lowercases text, normalizes quotation marks, removes punctuation except periods, percent signs, apostrophes, slashes, and hyphens, removes English articles, and collapses whitespace. Exact match compares the normalized prediction and reference. Soft match is the union of exact match, substring containment in either direction, and agreement between any extracted prediction/reference numbers within $\max(10^{-4}, 10^{-3}|a_{\mathrm{num}}|)$. Numeric extraction removes

thousands separators. These binary metrics support paired comparisons within the runner and differ from official VQA consensus, ANLS, and benchmark-specific numeric scores. AI2D predictions include option prefixes such as "B. Fish population would decrease." while references contain answer text; containment can accept the answer while exact match rejects the prefix. This accounts for the zero exact-match column in that workload.

The cell mapper first detects counting by "how many", "number of", or "count". Such requests enter `infographic_count` or `vqav2_count` according to the workload identifier. Among remaining requests, those beginning with "is", "are", "was", "were", "does", "do", "can", "has", "have", or "would" enter `binary_visual`. All others enter `default`. These rules and the minimum support of eight remain fixed during evaluation. The development cell sizes after deduplication are 12, 4, 29, and 402, respectively.

A.3 Combined Development and Held-Out Diagnostics

Route	Soft/Exact	Soft W/L	Exact W/L	Mean token/slot red. (%)
Fresh / Exact L1	573/354	0/6	10/0	0.00/0.00
KIVI-style proxy	541/329	20/58	25/40	0.00/–
SnapKV-style proxy	439/230	20/160	20/134	49.96/–
TokenPool	477/240	20/122	15/119	49.96/–
FacetGuard-base	579/344	–	–	5.13/2.57
VPCache	583/349	4/0	7/2	5.13/2.57

Table 8: Combined diagnostic over 896 requests, including development data. W/L uses FacetGuard-base. Reductions are request means over the entire matrix, including exact paths; slot reduction describes the selected memory representation.

The combined frozen policy assigns 59 requests to APVC-hifi, 33 to the KIVI-style view, and 804 to FacetGuard-base. These assignments differ from the count of shortened prefixes: guarded base paths also compress, and guarded APVC paths can retain Exact L1. Against the full-prefix reference, VPCache has soft W/L of 10/0 and exact W/L of 4/9. The corresponding base counts are 6/0 and 0/10.

Workload	Fresh	KIVI-style	SnapKV-style	TokenPool	Base	VPCache
ABO-MVR	20/3	17/3	12/0	16/1	20/3	20/3
AI2D	106/0	105/0	88/0	101/0	106/0	106/0
ChartQA	87/63	75/58	69/48	64/42	87/63	87/63
DocVQA	105/89	96/79	71/51	77/60	105/88	105/88
InfographicVQA	70/58	65/55	45/32	53/40	70/58	74/60
TextVQA	92/66	89/64	58/37	73/43	93/65	93/65
VQAv2	93/75	94/70	96/62	93/54	98/67	98/70

Table 9: Combined per-workload diagnostic, 128 requests per workload.

A.4 Full Frozen AP-CVS Policy

Table 7 reports the full frozen policy and its development counts. Removing APVC from the bank changes only the binary cell, which then retains the base. The static-bank selector in Table 6 uses this reduced policy. Both selectors use the same cell mapper, minimum support, and paired qualification rule.

A.5 Component Runs

Method	Soft	Exact	Avg. s	L2 use	Token red.	Mem. red.	Paired	Losses
Fresh	573	354	0.170	0	0.000	–	–	–
KIVI-2bit	541	329	0.430	592	0.000	–	–	–
SnapKV-window	439	230	0.439	592	0.499	–	–	–
TokenPool	477	240	0.475	592	0.499	–	–	–
FacetGuard-base	579	344	3.530	92	0.499	0.250	–	–
AP-CVS-only	583	346	3.986	125	0.364	0.182	+4/+2	0
APVC-hifi-only	579	347	0.144	92	0.499	0.250	+0/+3	0

Table 10: Raw component run summaries for audit. AP-CVS-only and APVC-only are not separate main methods, and these raw runs are not used as strict common-set comparisons in the main paper.

A.6 Per-Workload Visualization

Table 9 gives the combined numeric workload breakdown. The corresponding heatmap shows changes relative to fresh inference.

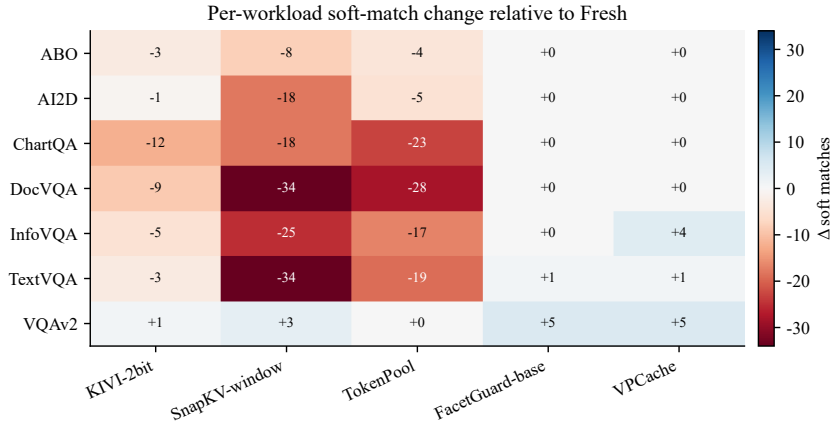


Figure 4: Per-workload soft-match changes relative to fresh inference. Static compression causes broad degradation, while VPCache concentrates gains in certified cells and keeps the remaining workloads on the base fallback.

A.7 Qualitative Method Comparison

Figure 5 compares the decoded answers on four examples from the combined request matrix. The infographic-count case shows complementary correctness: the KIVI-style view returns the reference count of six, whereas the full prefix and the other static views return five. The frozen counting route retains that benefit. For the binary visual question, APVC returns the exact answer "Yes"; the KIVI-style and TokenPool outputs include a trailing period and receive soft-only credit under the shared normalizer. This example distinguishes answer form from a change in the semantic yes/no decision.

The TextVQA and ChartQA cases illustrate the role of the guarded base. All three static views miss the book title or chart difference, while VPCache retains the correct base answer. Together, these cases show why a single compression view does not serve every question cell: a view useful for counting can lose text or numerical detail elsewhere. The selected route is fixed by the cell policy before the answer is generated.

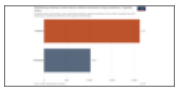
Input / query	Fresh	KIVI-2bit	SnapKV	TokenPool	VPCache
(a) InfoVQA  Q: How many types of waste can be recycled? **GT: 6**	5 ✗ Wrong	6 ● Exact	5 ✗ Wrong	5 ✗ Wrong	*KIVI view* 6 ● Exact
(b) VQAv2  Q: Is this a baby elephant? **GT: yes**	Yes ● Exact	Yes. ◆ Soft only	No ✗ Wrong	Yes. ◆ Soft only	*APVC view* Yes ● Exact
(c) TextVQA  Q: what is the title of the green book that is second from the bottom? **GT: the speaking eye**	The Speaking Eye ● Exact	The title of the green book second from the bottom is "The Body." ✗ Wrong	The title of the green book that is second from the bottom is "The Painted... ✗ Wrong	The title of the green book second from the bottom is "The Painted Veil." ✗ Wrong	*Base fallback* The Speaking Eye ● Exact
(d) ChartQA  Q: What is the difference in the value between Incidence and Prevalence? **GT: 1092**	1092 ● Exact	1122 ✗ Wrong	1120 ✗ Wrong	1200 ✗ Wrong	*Base fallback* 1092 ● Exact

Figure 5: Representative outputs from the combined request matrix. Green, amber, and red indicate exact, soft-only, and incorrect answers. AP-CVS uses the KIVI-style view for infographic counting (a), APVC for a binary visual question (b), and the guarded base for text and chart questions (c–d). KIVI and SnapKV denote the in-runner proxies.

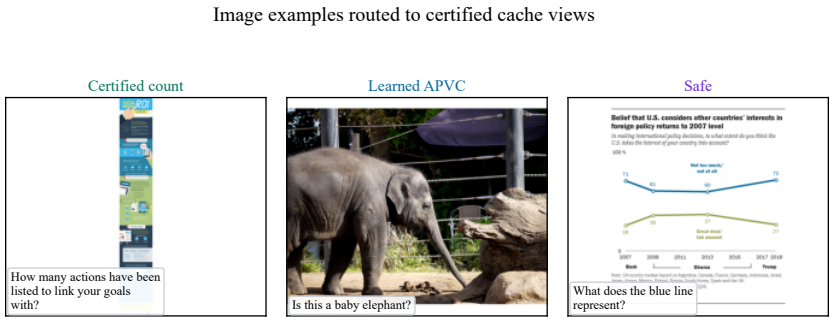


Figure 6: Additional visual inputs from the combined request matrix. The examples cover the certified infographic-count route, the learned APVC route, and the base-fallback route.

A.8 APVC Compressor Ablation

The ablation supports the high-fidelity APVC setting. The aggressive memory target loses OCR, numeric, and fine-detail evidence even with answer-weighted terms. Retaining more memory removes these paired losses; training at that target yields the zero-loss APVC-hifi component used by the selector. The strict variant retains the 0.50 memory ratio. The 0.50-to-0.75 variant trains at the smaller budget and uses the larger budget at inference.

Variant	Soft	Exact	Avg. s	L2 use	Token red.	Mem. red.	Paired	Losses
Deterministic base	579	344	3.530	92	0.499	0.250	–	–
APVC-v1	572	342	3.196	92	0.499	0.499	-7/-2	7
APVC-v2, 0.50	573	343	0.228	92	0.499	0.499	-6/-1	6
APVC-v2, strict	573	345	0.296	83	0.499	0.499	-6/+1	6
APVC-v2, 0.50 to 0.75	578	347	0.219	92	0.499	0.250	-1/+3	1
Full APVC-hifi	579	347	0.144	92	0.499	0.250	+0/+3	0

Table 11: APVC learned-compressor ablation on the final L128 matrix. Paired deltas and losses are measured against the deterministic multifacet cache. These configurations vary the training objective, memory budget, and policy. The runtime values are archival diagnostics from different cache states.

A.9 Cross-Setting Validation

Method	Soft	Exact	Avg. s	L2 use	Token red.	Mem. red.	Paired	Losses
maxp131072-base	516	281	0.127	92	0.500	0.250	–	–
maxp131072-APVC-hifi	518	283	0.127	92	0.500	0.250	+2/+2	1

Table 12: Cross-setting validation at `max_pixels=131072`. This lower image budget is harder than the main setting, but APVC-hifi preserves the positive held-out trend.

Table 5 separates development and held-out requests for this resolution check. The combined result has three paired wins and one loss for both soft and exact match; the loss occurs in the development split. As in the main evaluation, request-index pairing retains all 896 requests.

A.10 Peak Memory Diagnostic

The earlier warm-cache peak-memory diagnostic checks implementation behavior on the smaller L16 workload. It predates APVC-hifi and serves as an auxiliary diagnostic. CUDA peak memory is reset after loading Qwen2.5-VL-7B. The reported allocation is the additional evaluation-stage peak above the loaded model baseline.

Method	Soft	Avg. s	L2 use	Token red.	Peak alloc. MB	Cache MB
Fresh	0.652	0.164	0.000	0.000	93.2	0.0
Exact L1 cache	0.652	0.106	0.000	0.000	93.2	73.5
TokenPool proxy	0.518	0.122	0.000	0.500	93.2	36.9
SparseVLM proxy	0.473	0.116	0.000	0.500	93.2	36.9
KIVI proxy	0.589	0.117	0.000	0.000	93.2	73.5
Online guarded cache	0.652	0.119	0.152	0.499	89.1	81.0

Table 13: Warm-cache peak-memory diagnostic on seven workloads with 16 examples per workload. Cache size is the on-disk file-cache footprint.

The online cache path does not increase evaluation-stage peak GPU allocation in this small L16 run. Its on-disk footprint includes exact fallback entries and selected compressed layout-L2 entries, so the file-cache footprint provides no storage advantage.