SPECKV: LOSSLESS SPECULATIVE DECODING VIA DYNAMIC KV SELECTION FOR EFFICIENT INFERENCE OF LARGE VISION-LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large vision-language models (LVLMs) support increasingly complex multimodal tasks, but long visual inputs and growing textual histories make autoregressive decoding costly. While existing approaches reduce drafting costs through visual context compression, one-shot selection overlooks changing context requirements and information overlap across modalities. We analyze two characteristics of multimodal drafting: inter-step context heterogeneity and intra-step context redundancy. Our analysis reveals that context requirements shift across drafting rounds, while visual evidence can overlap with information already expressed in the textual context, limiting the effectiveness of fixed visual subsets. To address these challenges, we propose `SpecKV`, a training-free multimodal speculative decoding framework that dynamically selects visual and textual key–value (KV) entries under a shared drafting budget. `SpecKV` combines verification-guided context adaptation with text-conditioned visual complementarity, using the latest accepted target state to assess relevance and a relevance-weighted textual reference to estimate cross-modal overlap. The active context is refreshed after each verification round, while inactive entries remain available for reselection. Experiments on seven image and video understanding benchmarks across Qwen2.5-VL and LLaVA-OneVision demonstrate acceleration in both standard and self-speculative settings. With a draft KV budget of 512, `SpecKV` achieves up to $3.33\times$ decoding speedup on image tasks and $2.78\times$ on video tasks over vanilla autoregressive decoding, while preserving the target output distribution through exact verification with full context.

1 INTRODUCTION

Large vision-language models (LVLMs) extend large language models (LLMs) to multimodal understanding by incorporating visual representations alongside textual instructions (Bai et al., 2025; Li et al., 2024a). High-resolution images and long videos introduce many visual tokens (Wang et al., 2024b; Bai et al., 2025), while detailed responses progressively lengthen the textual context. Together, these tokens enlarge the key–value (KV) cache and increase GPU memory consumption. During autoregressive decoding, each new token attends to the preceding context and accesses its cached keys and values; repeatedly reading model parameters and the growing KV cache creates a memory-bandwidth bottleneck that increases end-to-end generation latency.

Speculative decoding (SD) accelerates generation through inexpensive drafting and parallel target verification, preserving the target output distribution through exact acceptance and correction (Leviathan et al., 2023; Chen et al., 2023). To reduce drafting costs, recent methods restrict the KV entries the drafter accesses while retaining full context for verification (Sun et al., 2024; Sadhukhan et al., 2024). For LVLMs, SpecVLM supports self-speculative decoding through visual pruning, allowing the same model to draft with a compact visual context and verify with the complete input (Ji et al., 2025). However, a compact context is useful only if it retains the information needed for accurate drafting. Fixing the visual selection before generation can miss changing evidence requirements and retain visual information subsequently expressed in text, potentially sacrificing draft acceptance despite reducing KV access. Figure 1 motivates our analysis from two complementary perspectives:

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.

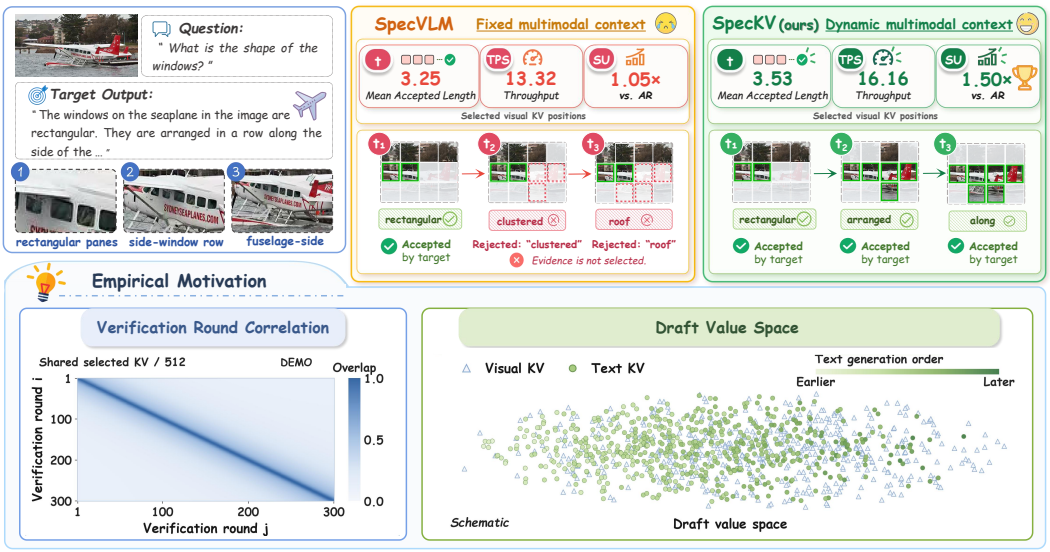

Figure 1: **Context dynamics and cross-modal redundancy. Top:** SpecVLM uses a fixed visual subset; SpecKV updates its context. Both use full-context target verification. **Bottom left:** overlap of 512-entry active KV lists over 300 rounds of one response, with 209 protected prompt entries. **Bottom right:** schematic cross-modal overlap as text grows, not a measured embedding.

(I) Inter-round context heterogeneity. The evidence required by the drafter can vary across rounds. In Figure 1 (top), the window description progresses from shape to arrangement and location, potentially requiring different visual regions. A subset selected only once cannot adapt to these changing requirements. The bottom-left panel tracks the selected context across 300 verification rounds of one response. Overlap between the complete 512-entry lists is 94.7% for adjacent rounds but 49.9% for rounds at least 100 apart. Excluding the 209 prompt entries protected by construction, overlap among the 303 reselected entries is 91.1% for adjacent rounds and zero between the first and last. This trace shows substantial long-range turnover in the selected context, motivating updates guided by the latest verification feedback rather than a fixed initial selection.

(II) Intra-round cross-modal redundancy. Within a round, visual and textual entries can represent the same evidence. Once the accepted prefix describes the windows as rectangular, both the corresponding visual region and its textual description may remain relevant. Retaining both can duplicate evidence and leave less budget for other relevant regions, such as those supporting arrangement or location. Entry-wise relevance ranks candidates independently, without accounting for overlap among retained entries. The bottom-right panel of Figure 1 schematically illustrates how accumulating text can cover information also encoded in the fixed image KV population. This motivates selecting visual context by both its relevance to the current continuation and its complementarity to retained text.

SpecKV is a training-free multimodal speculative decoding framework supporting standard and self-speculative settings. It combines verification-guided relevance with text-conditioned visual complementarity under a shared drafting budget, while retaining exact full-context target verification. In summary, our **main contributions** are as follows:

1. **Drafting Context Analysis.** We analyze multimodal drafting, identifying changing evidence requirements across rounds and potential visual–textual redundancy within rounds. These properties motivate verification-guided adaptation and complementarity-aware context selection.
2. **Mixing Relevance with Complementarity.** We propose SpecKV, combining verification-guided relevance with text-conditioned visual complementarity under a shared, text-prioritized KV budget. Context is refreshed after verification, with inactive entries retained for reselection.
3. **Comprehensive Experimental Validation.** Experiments on seven image and video benchmarks across Qwen2.5-VL and LLaVA-OneVision demonstrate faster standard and self-speculative decoding, achieving up to 3.33× speedup with exact full-context target verification.

2 Related work

Long-Context Vision–Language Models. Recent advances in large vision–language models (LVLMs) have expanded their ability to interpret complex visual inputs and generate detailed textual outputs (Bai et al., 2025; Li et al., 2024a). On the input side, high-resolution image and document understanding (Wang et al., 2024c; Mathew et al., 2020), multi-image reasoning (Jiang et al., 2024; Wang et al., 2024a), and long-video comprehension (Wu et al., 2024; Zhou et al., 2024) require preserving visual details across increasingly large inputs. On the output side, models such as LLaVA-CoT and Mulberry generate explicit intermediate reasoning steps before producing a final answer, extending the textual context (Xu et al., 2024; Yao et al., 2024). Detailed video captioning and summarization further motivate extended generation, as reflected in VDC and the generative tasks of MLVU (Chai et al., 2024; Zhou et al., 2024). These input and output demands jointly expand the KV cache, increasing memory consumption and the cost of attention computation and KV access during autoregressive decoding (Liu et al., 2026; Sun et al., 2024).

Long-Context Optimization. Longer contexts benefit LVLMs by supporting finer-grained perception and comprehensive multimodal understanding Wang et al. (2025b); Chen et al. (2025). Along the inference pipeline, efficient long-context processing methods address three stages: (i) *ViT encoding*, where token compression and merging reduce redundant visual representations before they enter the LLM, shortening the visual prefix but potentially discarding fine-grained evidence Yang et al. (2025); Shang et al. (2024); Li et al. (2024b); (ii) *LLM prefilling*, where attention- or diversity-guided token pruning Chen et al. (2024); Zhang et al. (2024); Xing et al. (2024) and sparse attention Li et al. (2025); Xu et al. (2025) reduce context-processing costs, though identifying the information needed for generation remains challenging; and (iii) *LLM decoding*, where KV cache pruning Li et al. (2024c); Wan et al. (2024); Wang et al. (2025a) and quantization Liu et al. (2024c) reduce storage and memory traffic by selecting entries or lowering their precision. These approaches improve long-context inference efficiency, but their gains depend on the compression level, and removing relevant information can alter the output distribution and impair generation quality.

Speculative Decoding. Speculative decoding (SD) uses a lightweight drafter to propose tokens for parallel target verification, amortizing autoregressive decoding costs Leviathan et al. (2023); Chen et al. (2023). However, long visual contexts can make drafting expensive. Existing methods include: (i) draft-training approaches that learn auxiliary predictors or decoding heads Cai et al. (2024); Li et al. (2024d), and (ii) training-free approaches using layer skipping or parallel draft–verify scheduling Zhang et al. (2023); Liu et al. (2024a). Multimodal extensions employ text-only drafters Gagrani et al. (2024), lightweight VLMs Huo et al. (2025), or twig branches Shao et al. (2025), balancing drafting cost against alignment with the target. Two approaches address visual context costs in video LLMs. Sparse-to-Dense (StD) Zhang et al. (2025) uses the same model to draft with sparse top-K attention and verify with dense attention, reducing drafting costs while preserving the target output distribution. SpecVLM Ji et al. (2025) combines verifier-guided importance selection with spatially uniform pruning to construct a compact visual context for self-speculative decoding. Overall, these methods reduce drafting costs through lightweight draft architectures or restricted context access, while retaining target verification to accept or correct the proposed tokens.

Unlike prior works that focus on visual token importance, we identify *cross-modal demand drift* in speculative decoding for LVLMs: context requirements change across drafting steps as generated text captures some visual information and subsequent steps require other evidence (Section 3.3). This motivates jointly selecting visual and textual KV entries under a shared budget and updating the selection throughout decoding to reflect the evolving generation context.

3 Methodology

3.1 Preliminaries

LVLM Inference. A large vision-language model (LVLM) generates text from visual and textual inputs. A vision encoder and projector map images or video frames to visual embeddings $\mathbf{F}^v \in \mathbb{R}^{N_v \times D}$, which together with textual embeddings $\mathbf{F}^t$ form $\mathbf{X} = [\mathbf{F}^v; \mathbf{F}^t]$ (Li et al., 2024a; Wang et al., 2024b). Here, N_v counts visual tokens and D is the embedding dimension. Prefilling constructs a layer-wise key–value (KV) cache over $\mathbf{X}$. We denote fixed visual KV positions by $\mathcal{V}$ and materialized

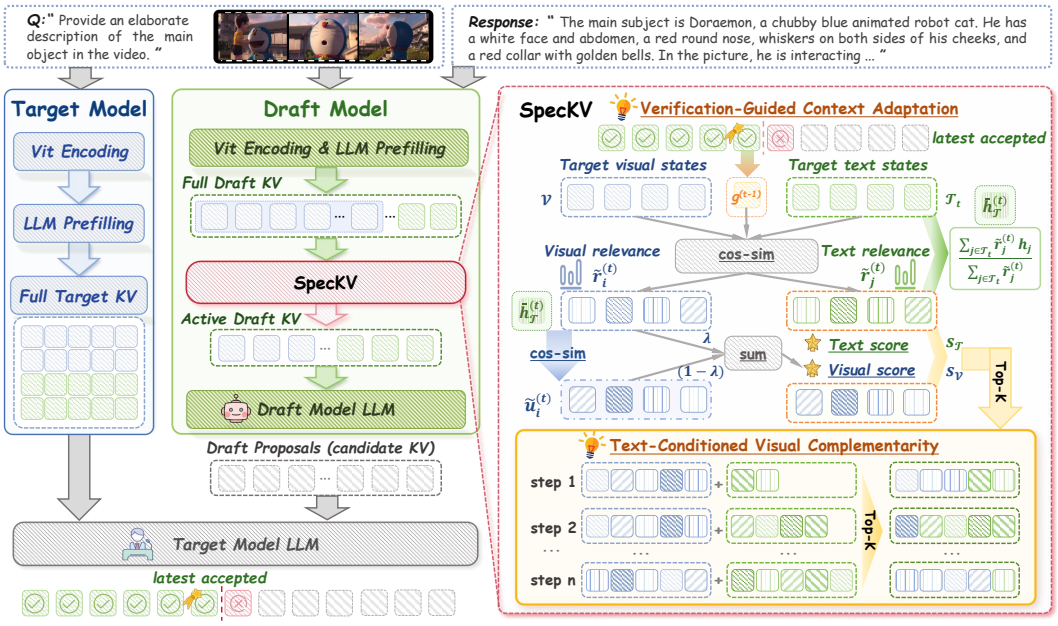

Figure 2: **Overview of SpecKV.** Verification-guided relevance and visual complementarity guide context selection under a text-prioritized budget B. Inactive entries remain available for reselection, while the target verifies with its full context.

textual KV positions before round t by $\mathcal{T}_t$, comprising prompt text and cached committed output. For target $\mathcal{M}_p$ and output $\mathbf{Y} = (y_1, \ldots, y_L)$, $p(\mathbf{Y} \mid \mathbf{X}) = \prod_j p(y_j \mid \mathbf{X}, \mathbf{Y}_{<j})$.

Speculative Decoding. Speculative decoding pairs $\mathcal{M}_p$ with a cheaper drafter $\mathcal{M}_q$ to reduce sequential target evaluations (Leviathan et al., 2023; Chen et al., 2023). For a stochastic draft chain of length γ, the target verifies proposals in parallel. At each reached position, acceptance uses $\min(1, p/q)$, where p is the target probability and q the actual probability used to generate that proposal at the same prefix. The first rejection triggers sampling from the normalized residual $[p - q]_+$; full acceptance yields an additional target sample. With full-context target probabilities, this rule preserves the target distribution in exact arithmetic. Our experiments use greedy verification, following target-argmax matches with branch-causal masks for candidate trees. We denote mean accepted draft tokens per round by τ, excluding correction and bonus tokens.

3.2 Overview of SpecKV

Problem Formulation. We consider full-context target verification with restricted historical context for drafting. Before round t, we select $\mathcal{I}_t \subseteq \mathcal{V} \cup \mathcal{T}_t$ with $|\mathcal{I}_t| \leq B$ from materialized draft KV entries. The drafter attends to this history and preceding tokens on its proposed path; temporary proposal KV is additional to B. Let $\bar{C}$ denote mean round time, including drafting, verification, selection, and cache updates, and $\bar{g}$ mean committed output tokens. Latency is $\bar{C}/\bar{g}$ (Appendix A.1.3). Lowering B can reduce drafting cost but also reduce $\bar{g}$ when necessary evidence is omitted. Selection should therefore track changing evidence, avoid redundancy, and remain inexpensive.

Overview. SpecKV (Figure 2) introduces two components in the drafting–verification loop. *Verification-guided context adaptation* (Section 3.3) uses accepted target states to refresh candidate relevance without an extra target forward pass. *Text-conditioned visual complementarity* (Section 3.4) estimates visual divergence from relevant text. Section 3.5 specifies text-prioritized allocation under the shared budget; Appendix A.3 details the decoding and cache operations.

3.3 Verification-Guided Context Adaptation

Evidence requirements can change between rounds, making one-time relevance estimates stale. We refresh relevance using full-context target states already produced by verification. These states summarize accepted progress and provide a selection proxy without an additional target forward pass.

Let $\mathbf{g}^{(t)} \in \mathbb{R}^d$ denote the final-layer target state after consuming the latest accepted draft token before round t, and $\mathbf{h}_i$ the corresponding target-space descriptor of candidate $i \in \mathcal{V} \cup \mathcal{T}_t$. Descriptors come from the same target layer, cached during prefilling for prompt positions and when committed text is processed. We define the relevance proxy as

$$r_i^{(t)} = \cos\Big(\mathbf{g}^{(t)}, \mathbf{h}_i\Big), \qquad \tilde{r}_i^{(t)} = \mathcal{N}\Big(r^{(t)}\Big)_i, \tag{1}$$

where $\mathcal{N}$ is min–max normalization, $\mathcal{N}(\mathbf{z})_i = (z_i - \min_j z_j)/(\max_j z_j - \min_j z_j + \epsilon)$ with $\epsilon > 0$, applied separately within $\mathcal{V}$ and $\mathcal{T}_t$. Cosine measures directional agreement, while normalization preserves within-modality rankings without calibrating scores across modalities. Pre-normalized descriptors permit relevance scoring with one matrix–vector product.

The reference is initialized with the final prompt state and updated to the state after the last accepted draft token; if none is accepted, the previous reference is retained. For differing visual grids in Standard-SD, $\mathbf{h}_i$ denotes a coordinate-overlap-weighted aggregate of target states aligned with draft position i. All comparisons remain in target space rather than mixing target and draft representations.

3.4 Text-Conditioned Visual Complementarity

Independent relevance scores can favor overlapping evidence. Once an image-derived fact appears in the accepted prefix, both its visual source and textual description may remain relevant. Retaining both may add less information than preserving other relevant visual evidence not yet expressed in text. We therefore complement relevance with a text-conditioned estimate of visual redundancy.

Comparing all visual–textual candidates costs $O(|\mathcal{V}|\,|\mathcal{T}_t|\,d)$ per round. Instead, we aggregate relevant text into one reference. For positive total textual relevance, define

$$\bar{\mathbf{h}}_{\mathcal{T}}^{(t)} = \frac{\sum_{j\in\mathcal{T}_t} \tilde{r}_j^{(t)} \mathbf{h}_j}{\sum_{j\in\mathcal{T}_t} \tilde{r}_j^{(t)}}, \qquad u_i^{(t)} = 1 - \cos\Big(\mathbf{h}_i, \bar{\mathbf{h}}_{\mathcal{T}}^{(t)}\Big), \tag{2}$$

with $\tilde{u}_i^{(t)} = \mathcal{N}(u^{(t)})_i$ normalized within $\mathcal{V}$. A large $u_i^{(t)}$ indicates directional divergence from relevant text, not measured information gain or a causal contribution to acceptance. Updating the reference with $\tilde{r}^{(t)}$ makes this complementarity proxy evolve with the generated response.

Relevance measures agreement with the current continuation, whereas complementarity favors visual candidates distinct from relevant text. We combine the two visual scoring terms as

$$s_i^{(t)} = \lambda\, \tilde{r}_i^{(t)} + (1-\lambda)\, \tilde{u}_i^{(t)}, \qquad i \in \mathcal{V}, \tag{3}$$

where $\lambda \in [0,1]$ controls the trade-off and the endpoints retain one visual scoring term. We use $\lambda = 0.25$ by default (Appendix A.4). If nonempty text has zero total relevance, we explicitly replace the weighted reference with the unweighted mean of its descriptors. With no text or a zero-norm reference, we set $s_i^{(t)} = \tilde{r}_i^{(t)}$ directly. These operations change scores, not the textual KV entries.

3.5 Budget Allocation

Because scores are normalized within each modality, we allocate slots before ranking rather than applying a global Top-B. We prioritize textual context and assign the remaining slots to visual candidates using their combined scores:

$$b_T = \min\{|\mathcal{T}_t|, B\}, \qquad b_V = \min\{B - b_T, |\mathcal{V}|\}, \qquad \mathcal{I}_t = \mathcal{T}_t^{(b_T)} \cup \mathrm{Top}_{b_V}(s^{(t)}). \tag{4}$$

Here, $\mathcal{T}_t^{(b_T)}$ contains the b_T textual positions with highest $\tilde{r}_i^{(t)}$, and Top_{b_V} returns the b_V highest-scoring visual positions. When $|\mathcal{T}_t| \le B$, all available text is retained; when $|\mathcal{T}_t| \ge B$, $b_V = 0$. Thus, whenever visual slots remain, the reference in Equation 2 summarizes retained text.

The selected indices gather draft-native KV entries, $\mathcal{C}_q^{(t)} = (\mathbf{K}_q^{(t)}[\mathcal{I}_t], \mathbf{V}_q^{(t)}[\mathcal{I}_t])$, with original positions preserved for rotary and multimodal positional encodings (Bai et al., 2025). Target descriptors score positions; they do not replace draft KV. Selection changes only the active view, leaving inactive bank entries available for later reselection without recomputation.

Descriptor scoring costs $O(N_c d)$ per refresh at one target layer, where $N_c = |\mathcal{V}| + |\mathcal{T}_t|$. Index selection and KV gathering incur additional costs included in $\bar{C}$. The budget bounds active historical KV access, not total resident cache storage. Exact verification preserves the target distribution for stochastic chains and the target's greedy sequence for greedy decoding under the respective assumptions in Appendix A.1, which also gives the net-benefit condition on $\bar{C}/\bar{g}$.

Net Benefit of Context Adaptation. For static and dynamic selectors producing the same output sequence, let $(\bar{C}_m, \bar{g}_m)$ denote positive mean round time and committed token count for $m \in \{\mathrm{stat}, \mathrm{dyn}\}$. Since latency per token is $\bar{C}_m/\bar{g}_m$, dynamic selection is faster if and only if

$$\frac{\bar{g}_{\mathrm{dyn}}}{\bar{g}_{\mathrm{stat}}} > \frac{\bar{C}_{\mathrm{dyn}}}{\bar{C}_{\mathrm{stat}}}.$$

When adaptation increases round cost, its relative gain in committed tokens must be larger. Conversely, lower round cost can yield a speedup without a longer accepted prefix. All selection and cache-update costs are included in $\bar{C}_m$; Appendix A.1.3 provides the derivation and cost definitions.

4 Experiments

4.1 Experimental Setting

Model Pairing Details. We evaluate two LVLM families, Qwen2.5-VL (Bai et al., 2025) and LLaVA-OneVision (Li et al., 2024a), in two decoding settings. (i) **Standard-SD (Std.-SD)** (Leviathan et al., 2023; Chen et al., 2023) pairs larger targets with smaller drafters from the same family: Qwen2.5-VL-32B/7B and LLaVA-OneVision-72B/7B. (ii) **Self-SD** uses the same model for drafting and verification, with restricted drafting and full verification contexts. We use Qwen2.5-VL-7B and LLaVA-OneVision-7B in this setting. Neither requires additional training.

Benchmark Details. We evaluate seven multimodal benchmarks. Image tasks include OCR-Bench (Liu et al., 2024b), MMSpec (Shen et al., 2026), MathVista (Lu et al., 2024), and ScienceQA (Lu et al., 2022). Video tasks include Video Detail Caption (VDC) (Chai et al., 2024) and Video Detail Description (VDD) (LMMs-Lab, 2024) for detailed captioning, and MovieChat (Song et al., 2023) for video question answering. Benchmark details appear in Appendix A.2.1.

Baseline Details. We compare `SpecKV` with vanilla autoregressive decoding (Vanilla), SD-Tree, SD-Rand, Sparse-to-Dense (StD) (Zhang et al., 2025), and SpecVLM (Ji et al., 2025). These baselines cover full-context speculative drafting, random visual selection, sparse attention, and verifier-guided visual pruning. Tree-based methods use fixed candidate trees with a maximum drafting depth of five. Appendix A.2.2 describes these methods and their evaluation configurations.

Implementation Details. Unless varied in an ablation, `SpecKV` uses a historical draft KV budget of $B = 512$ and a relevance weight of $\lambda = 0.25$. All evaluations use greedy decoding and terminate at EOS or a maximum of 2048 newly generated tokens for image benchmarks and 512 for video benchmarks, with no minimum output length. The same generation limit applies to all methods on each benchmark and is independent of the historical KV budget. Experiments run on two NVIDIA H200 GPUs; implementation and evaluation details are provided in Appendix A.2.4.

Evaluation Metrics. We report decode-only throughput (tokens/s), speedup over Vanilla with the same target checkpoint, and mean accepted draft length τ, excluding root, correction, and bonus tokens. Timing starts after prefilling and the first target token, and includes subsequent drafting, context selection, target verification, and cache updates. Visual encoding and prefilling are excluded. Throughput divides total generated tokens in the timed segment by total decoding time across requests; speedup compares decoding times on the same requests. "Avg." is the arithmetic mean of benchmark-level speedups. The aggregation of τ is detailed in Appendix A.2.4.

Table 1: Speedup ratios and mean accepted length τ on image understanding tasks.

Method	OCRBench			MMSpec			MathVista			ScienceQA			Avg.
	τ	Tokens/s	Speedup	τ	Tokens/s	Speedup	τ	Tokens/s	Speedup	τ	Tokens/s	Speedup	Speedup
Qwen2.5-VL (Standard-SD; 32B–7B)													
Vanilla	—	10.83	1.00	—	11.09	1.00	—	11.53	1.00	—	11.29	1.00	1.00
SD-Tree	3.20	13.97	1.29	3.14	14.90	1.34	4.27	19.99	1.73	3.97	18.75	1.66	1.51
SD-Rand	3.10	17.32	1.60	3.12	17.19	1.55	4.23	22.32	1.94	3.96	20.72	1.84	1.73
StD	2.17	17.28	1.60	2.16	18.17	1.64	2.88	23.22	2.01	2.69	22.10	1.96	1.80
SpecVLM	3.13	17.68	1.63	3.10	17.13	1.55	4.24	22.19	1.92	3.96	20.57	1.82	1.73
SpecKV	**3.09**	**20.31**	**1.88**	**3.08**	**21.36**	**1.93**	**4.11**	**27.30**	**2.37**	**3.84**	**25.98**	**2.30**	**2.12**
Self-SD Qwen2.5-VL 7B–7B													
Vanilla	—	22.43	1.00	—	24.74	1.00	—	24.75	1.00	—	27.68	1.00	1.00
SD-Tree	4.66	20.63	0.92	4.72	21.31	0.86	4.90	24.04	0.97	4.91	25.62	0.93	0.92
SD-Rand	4.08	24.28	1.08	4.20	25.52	1.03	4.51	26.16	1.06	4.71	27.73	1.00	1.04
StD	4.39	33.06	1.47	4.16	32.21	1.30	4.77	35.86	1.45	4.76	38.09	1.38	1.40
SpecVLM	3.58	21.81	0.97	3.54	22.40	0.91	4.17	24.95	1.01	4.41	26.19	0.95	0.96
SpecKV	**4.06**	**30.03**	**1.34**	**3.99**	**31.24**	**1.26**	**4.57**	**37.01**	**1.50**	**4.64**	**38.43**	**1.39**	**1.37**
Standard-SD LLaVA-OneVision 72B–7B													
Vanilla	—	6.12	1.00	—	6.67	1.00	—	6.63	1.00	—	6.77	1.00	1.00
SD-Tree	2.51	9.70	1.59	2.47	9.60	1.44	3.71	13.94	2.10	3.13	12.83	1.89	1.76
SD-Rand	2.49	11.61	1.90	2.45	11.49	1.72	3.58	16.45	2.48	3.20	15.57	2.30	2.10
StD	1.77	12.69	2.07	1.92	12.61	1.89	3.13	18.47	2.79	2.48	16.75	2.47	2.31
SpecVLM	2.53	11.76	1.92	2.49	11.63	1.74	3.75	16.94	2.56	3.16	15.49	2.29	2.13
SpecKV	**2.49**	**15.30**	**2.50**	**2.45**	**15.14**	**2.27**	**3.62**	**22.09**	**3.33**	**3.16**	**20.11**	**2.97**	**2.77**
Self-SD LLaVA-OneVision 7B–7B													
Vanilla	—	30.30	1.00	—	35.53	1.00	—	29.31	1.00	—	30.43	1.00	1.00
SD-Tree	4.20	25.48	0.84	4.12	30.45	0.86	4.47	26.87	0.92	4.35	26.74	0.88	0.87
SD-Rand	3.66	39.86	1.32	3.80	42.78	1.20	4.30	36.53	1.25	4.04	34.90	1.15	1.23
StD	4.19	45.05	1.49	4.24	52.29	1.47	4.68	42.48	1.45	4.54	48.47	1.59	1.50
SpecVLM	4.05	41.86	1.38	3.99	43.98	1.24	4.40	37.40	1.28	4.14	35.26	1.16	1.26
SpecKV	**3.93**	**48.98**	**1.62**	**4.06**	**53.82**	**1.51**	**4.40**	**54.31**	**1.85**	**4.10**	**50.58**	**1.66**	**1.66**

4.2 Main Results

Decoding Efficiency on Image Understanding Benchmarks. Table 1 evaluates four benchmarks across both model families and drafting settings, yielding three observations. (i) **Broad acceleration:** SpecKV improves over Vanilla in every configuration, with speedups from $1.26\times$ to $3.33\times$. (ii) **Consistent improvements under Std.-SD:** SpecKV achieves the highest speedup on every benchmark for both families, averaging $2.12\times$ on Qwen2.5-VL and $2.77\times$ on LLaVA-OneVision, compared with $1.80\times$ and $2.31\times$ for StD. (iii) **Applicability to Self-SD:** SpecKV surpasses SpecVLM on all four benchmarks for both families, with average speedups of $1.37\times$ and $1.66\times$. StD remains slightly faster on Qwen2.5-VL on average ($1.40\times$), whereas SpecKV outperforms StD on LLaVA-OneVision ($1.66\times$ versus $1.50\times$). Thus, gains extend to both settings but vary by model family.

Decoding Efficiency on Video Understanding Benchmarks. Table 2 shows that SpecKV achieves the highest speedup in eleven of twelve model–setting–benchmark configurations. Under Std.-SD, average speedups reach $2.26\times$ for Qwen2.5-VL and $2.47\times$ for LLaVA-OneVision. Under Self-SD, SpecKV outperforms all baselines on every video benchmark, averaging $2.54\times$ and $1.63\times$, compared with $2.37\times$ and $1.49\times$ for StD. The only exception to the overall ranking is MovieChat under Std.-SD with Qwen2.5-VL, where StD is slightly faster ($2.00\times$ versus $1.98\times$). These results extend the gains to detailed video captioning and question answering, with consistent improvements over SpecVLM across all evaluated model families and both drafting settings.

Table 2: Speedup ratios and mean accepted length τ on video captioning and video question answering tasks.

Method	VDC			VDD			MovieChat			Avg.
	τ	Tokens/s	Speedup	τ	Tokens/s	Speedup	τ	Tokens/s	Speedup	Speedup
Standard-SD Qwen2.5-VL 32B–7B										
Vanilla	—	7.95	1.00	—	10.07	1.00	—	9.81	1.00	1.00
SD-Tree	3.27	10.87	1.37	3.35	12.12	1.20	3.23	11.67	1.19	1.25
SD-Rand	3.21	15.80	1.99	3.29	17.62	1.75	3.17	16.97	1.73	1.82
StD	2.36	18.24	2.29	2.38	20.33	2.02	2.29	19.58	2.00	2.10
SpecVLM	3.29	15.90	2.00	3.31	17.96	1.76	3.19	17.05	1.74	1.83
SpecKV	**3.24**	**21.70**	**2.73**	**3.24**	**20.95**	**2.08**	**3.03**	**19.43**	**1.98**	**2.26**
Self-SD Qwen2.5-VL 7B–7B										
Vanilla	—	20.60	1.00	—	24.75	1.00	—	25.51	1.00	1.00
SD-Tree	4.38	17.49	0.85	4.45	19.36	0.78	4.19	17.83	0.70	0.78
SD-Rand	3.75	30.49	1.48	3.81	33.76	1.36	3.58	31.09	1.22	1.35
StD	3.64	53.41	2.59	3.67	59.14	2.39	3.46	54.47	2.14	2.37
SpecVLM	3.85	31.45	1.53	3.89	35.00	1.41	3.66	32.20	1.26	1.40
SpecKV	**4.02**	**57.27**	**2.78**	**3.81**	**63.41**	**2.56**	**4.02**	**58.40**	**2.29**	**2.54**
Standard-SD LLaVA-OneVision 72B–7B										
Vanilla	—	6.62	1.00	—	6.00	1.00	—	4.41	1.00	1.00
SD-Tree	3.68	12.24	1.85	3.73	11.60	1.93	3.99	6.58	1.49	1.76
SD-Rand	3.19	14.72	2.22	3.24	13.95	2.33	3.46	7.91	1.79	2.11
StD	2.87	15.02	2.27	3.01	14.25	2.37	3.21	8.07	1.83	2.16
SpecVLM	3.37	15.75	2.38	3.53	14.94	2.49	3.77	8.45	1.92	2.26
SpecKV	**3.44**	**17.18**	**2.59**	**3.33**	**16.29**	**2.71**	**2.26**	**9.23**	**2.09**	**2.47**
Self-SD LLaVA-OneVision 7B–7B										
Vanilla	—	26.45	1.00	—	17.66	1.00	—	17.23	1.00	1.00
SD-Tree	4.50	23.82	0.90	4.47	19.58	1.11	4.71	14.24	0.83	0.95
SD-Rand	3.81	33.30	1.26	3.78	27.38	1.55	3.99	19.91	1.16	1.32
StD	3.72	37.60	1.42	3.76	30.91	1.75	3.97	22.48	1.30	1.49
SpecVLM	3.91	35.46	1.34	3.95	29.08	1.65	4.17	21.19	1.23	1.41
SpecKV	**3.87**	**41.06**	**1.55**	**3.79**	**33.76**	**1.91**	**3.03**	**24.55**	**1.42**	**1.63**

4.3 Ablation Studies

We examine compression scope, context refresh, scoring, and KV budget with Qwen2.5-VL-32B/7B under Std.-SD. Table 3 compares scope and update policy on VDC, VDD, MathVista, and OCRBench; Figures 3 and 4 examine scoring, budget, and runtime on VDC. Unless varied, $B = 512$ and $\lambda = 0.25$. Appendix A.4 details the experimental settings and controlled comparisons.

Table 3: Compression scope and selection update policy on four benchmarks.

Configuration	VDC			VDD			MathVista			OCRBench			Avg.
	τ	Tokens/s	Speedup	τ	Tokens/s	Speedup	τ	Tokens/s	Speedup	τ	Tokens/s	Speedup	Speedup
Reference (uncompressed historical KV)													
AR (full context)	—	7.95	1.00	—	10.07	1.00	—	11.53	1.00	—	10.83	1.00	1.00
Full-context SD	3.55	13.00	1.64	3.54	12.55	1.25	4.51	16.36	1.42	3.39	12.17	1.12	1.36
Compression scope (one modality left uncompressed)													
Text-only	3.48	12.72	1.60	3.48	12.28	1.22	4.42	16.00	1.39	3.32	11.90	1.10	1.33
Visual-only	3.32	20.11	2.53	3.31	18.93	1.88	4.21	25.02	2.17	3.16	18.19	1.68	2.06
Update policy ($B = 512$, text-prioritized allocation; Eq. 4)													
Static SpecKV	3.20	20.75	2.61	3.20	19.54	1.94	4.06	25.83	2.24	3.05	19.17	1.77	2.14
SpecKV (dynamic)	**3.24**	**21.70**	**2.73**	**3.24**	**20.95**	**2.08**	**4.11**	**27.30**	**2.37**	**3.09**	**20.31**	**1.88**	**2.27**

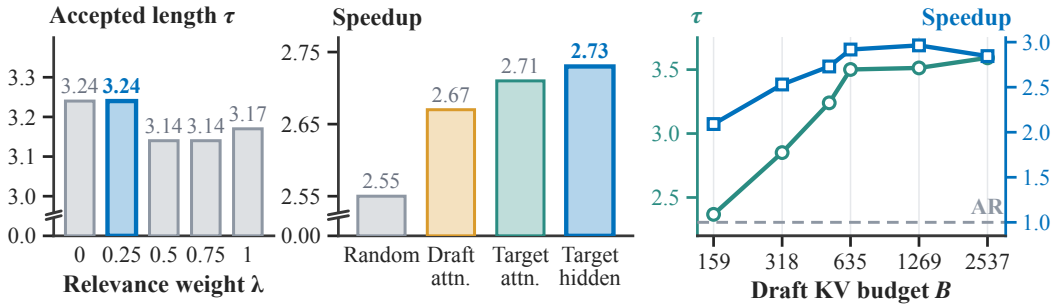

Figure 3: **Scoring and budget ablations** on VDC. Left/center: $B = 512$; center: no complementarity. Bar axes use breaks; AR denotes autoregressive decoding. Results are from single runs.

Compression Scope. Visual-only compression is faster than text-only compression on all four benchmarks, averaging $2.06\times$ versus $1.33\times$, compared with $1.36\times$ for full-context SD (Table 3). Visual compression therefore provides most of the observed gain. These variants retain the other modality in full and do not constitute an equal-total-budget comparison.

Hyperparameter Sensitivity. Varying $\lambda \in [0, 1]$ yields $\tau = 3.14$–3.24, with our default $\lambda = 0.25$ matching the highest acceptance in this sweep. Complementarity-only ranking still uses relevance-weighted text. This comparison does not establish a universally optimal mixture. Increasing B from 159 to 2537 raises τ from 2.37 to 3.59, but speedup peaks at $2.96\times$ for $B = 1269$ before falling to $2.85\times$ (Figure 3, left and right). More context thus trades higher acceptance for greater drafting cost. Our common $B = 512$ yields $2.73\times$ and limits historical KV access, not response length.

Relevance Signal. Without complementarity, target hidden-state cosine and attention reach $2.73\times$ and $2.71\times$, versus $2.67\times$ for draft attention and $2.55\times$ for random ranking (Figure 3, center). Hidden states offer comparable throughput, and random selection also benefits from restricted context. Single runs do not establish a statistically reliable ranking between the target signals.

Dynamic Context Update. At $B = 512$, both selectors use the same text-prioritized allocation. Updating the context improves τ and throughput on all four benchmarks (Table 3), raising average speedup from $2.14\times$ to $2.27\times$. Full-context SD has higher acceptance but remains slower. The additional accepted tokens do not fully offset the greater cost of processing the complete visual and textual history throughout the drafting process.

The VDC profile (Figure 4) illustrates this cost–progress trade-off. Dynamic selection reduces drafting time from 21.0 to 18.1 ms/token and total latency from 48.19 to 46.08 ms/token. The 4.4% net reduction includes scoring, selection, and cache updates; the drafting savings therefore outweigh adaptation overhead in this configuration.

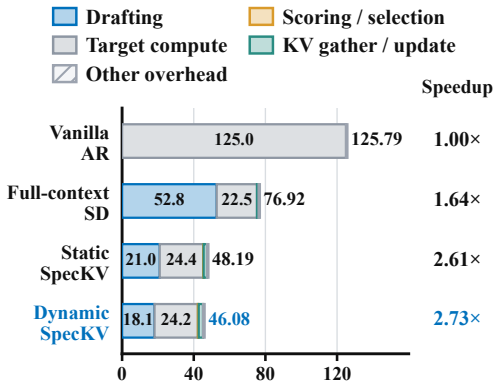

Figure 4: **Runtime breakdown** .

5 Conclusion

We present SpecKV, a training-free framework that addresses changing evidence requirements and visual–textual redundancy in multimodal speculative decoding. It combines verification-guided relevance and text-conditioned visual complementarity under a shared, text-prioritized budget, refreshing the active context while retaining inactive entries for reuse. Experiments on seven benchmarks and two LVLM families demonstrate acceleration in standard and self-speculative settings. The ablations and runtime analysis highlight the need to balance accepted progress against drafting and context-update costs, rather than maximizing acceptance alone. Full-context target verification preserves the target's generation behavior under the stated acceptance and correction rules.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript drafting, language polishing, and structural editing, as well as with literature search and code development.

REFERENCES

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, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025. URL https://arxiv.org/abs/2502.13923.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 5209–5235. PMLR, 2024. URL https://proceedings.mlr.press/v235/cai24b.html.

Wenhao Chai, Enxin Song, Yilun Du, Chenlin Meng, Vashisht Madhavan, Omer Bar-Tal, Jenq-Neng Hwang, Saining Xie, and Christopher D. Manning. AuroraCap: Efficient, Performant Video Detailed Captioning and a New Benchmark. *arXiv preprint arXiv:2410.03051*, 2024. URL https://arxiv.org/abs/2410.03051.

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

Guo Chen, Zhiqi Li, Shihao Wang, Jindong Jiang, Yicheng Liu, Lidong Lu, De-An Huang, Wonmin Byeon, Matthieu Le, Tuomas Rintamaki, Tyler Poon, Max Ehrlich, Tong Lu, Limin Wang, Bryan Catanzaro, Jan Kautz, Andrew Tao, Zhiding Yu, and Guilin Liu. Eagle 2.5: Boosting Long-Context Post-Training for Frontier Vision-Language Models. *arXiv preprint arXiv:2504.15271*, 2025. URL https://arxiv.org/abs/2504.15271.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, 2024. URL https://arxiv.org/abs/2403.06764.

Mukul Gagrani, Raghavv Goel, Wonseok Jeon, Junyoung Park, Mingu Lee, and Christopher Lott. On speculative decoding for multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 8285–8289, 2024. URL https://openaccess.thecvf.com/content/CVPR2024W/ELVM/html/Gagrani_On_Speculative_Decoding_for_Multimodal_Large_Language_Models_CVPRW_2024_paper.html.

Mingxiao Huo, Jiayi Zhang, Hewei Wang, Jinfeng Xu, Zheyu Chen, Huilin Tai, and Yijun Chen. Spec-LLaVA: Accelerating Vision-Language Models with Dynamic Tree-Based Speculative Decoding. *arXiv preprint arXiv:2509.11961*, 2025. URL https://arxiv.org/abs/2509.11961.

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

Dongfu Jiang, Xuan He, Huaye Zeng, Cong Wei, Max Ku, Qian Liu, and Wenhu Chen. MANTIS: Interleaved Multi-Image Instruction Tuning. *arXiv preprint arXiv:2405.01483*, 2024. URL https://arxiv.org/abs/2405.01483.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-OneVision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024a. URL https://arxiv.org/abs/2408.03326.

Wentong Li, Yuqian Yuan, Jian Liu, Dongqi Tang, Song Wang, Jie Qin, Jianke Zhu, and Lei Zhang. TokenPacker: Efficient Visual Projector for Multimodal LLM. *arXiv preprint arXiv:2407.02392*, 2024b. URL https://arxiv.org/abs/2407.02392.

Yucheng Li, Huiqiang Jiang, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Amir H. Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, and Lili Qiu. MMInference: Accelerating Pre-filling for Long-Context VLMs via Modality-Aware Permutation Sparse Attention. *arXiv preprint arXiv:2504.16083*, 2025. URL https://arxiv.org/abs/2504.16083.

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. *arXiv preprint arXiv:2404.14469*, 2024c. URL https://arxiv.org/abs/2404.14469.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024d. URL https://arxiv.org/abs/2401.15077.

Tianyu Liu, Yun Li, Qitan Lv, Kai Liu, Jianchen Zhu, Winston Hu, and Xiao Sun. PEARL: Parallel Speculative Decoding with Adaptive Draft Length. *arXiv preprint arXiv:2408.11850*, 2024a. URL https://arxiv.org/abs/2408.11850.

Xuyang Liu, Xiyan Gui, Yuchao Zhang, and Linfeng Zhang. Mixing importance with diversity: Joint optimization for KV cache compression in large vision-language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.20707.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xucheng Yin, Cheng-lin Liu, Lianwen Jin, and Xiang Bai. OCRBench: On the hidden mystery of OCR in large multimodal models. *Science China Information Sciences*, 67(12):220102, 2024b. doi: 10.1007/s11432-024-4235-6. URL https://arxiv.org/abs/2305.07895.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. *arXiv preprint arXiv:2402.02750*, 2024c. doi: 10.48550/arXiv.2402.02750. URL https://arxiv.org/abs/2402.02750.

LMMs-Lab. Video detail caption. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/lmms-lab/VideoDetailCaption.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2209.09513.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.02255.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A Dataset for VQA on Document Images. *arXiv preprint arXiv:2007.00398*, 2020. URL https://arxiv.org/abs/2007.00398.

Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. MagicDec: Breaking the Latency-Throughput Tradeoff for Long Context Generation with Speculative Decoding. *arXiv preprint arXiv:2408.11049*, 2024. URL https://arxiv.org/abs/2408.11049.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. LLaVA-PruMerge: Adaptive Token Reduction for Efficient Large Multimodal Models. *arXiv preprint arXiv:2403.15388*, 2024. URL https://arxiv.org/abs/2403.15388.

Zhenwei Shao, Mingyang Wang, Zhou Yu, Wenwen Pan, Yan Yang, Tao Wei, Hongyuan Zhang, Ning Mao, Wei Chen, and Jun Yu. Growing a twig to accelerate large vision-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20064–20074, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Shao_Growing_a_Twig_to_Accelerate_Large_Vision-Language_Models_ICCV_2025_paper.html.

Hui Shen, Xin Wang, Ping Zhang, Yunta Hsieh, Qi Han, Zhongwei Wan, Ziheng Zhang, Jingxuan Zhang, Jing Xiong, Ziyuan Liu, Yifan Zhang, Hangrui Cao, Chenyang Zhao, and Mi Zhang. MMSpec: Benchmarking speculative decoding for vision-language models. *arXiv preprint arXiv:2603.14989*, 2026. URL https://arxiv.org/abs/2603.14989.

Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, Yan Lu, Jenq-Neng Hwang, and Gaoang Wang. MovieChat: From dense token to sparse memory for long video understanding. *arXiv preprint arXiv:2307.16449*, 2023. URL https://arxiv.org/abs/2307.16449.

Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. TriForce: Lossless acceleration of long sequence generation with hierarchical speculative decoding. *arXiv preprint arXiv:2404.11912*, 2024. URL https://arxiv.org/abs/2404.11912.

Zhongwei Wan, Ziang Wu, Che Liu, Jinfa Huang, Zhihong Zhu, Peng Jin, Longyue Wang, and Li Yuan. LOOK-M: Look-Once Optimization in KV Cache for Efficient Multimodal Long-Context Inference. *arXiv preprint arXiv:2406.18139*, 2024. URL https://arxiv.org/abs/2406.18139.

Fei Wang, Xingyu Fu, James Y. Huang, Zekun Li, Qin Liu, Xiaogeng Liu, Mingyu Derek Ma, Nan Xu, Wenxuan Zhou, Kai Zhang, Tianyi Lorena Yan, Wenjie Jacky Mo, Hsiang-Hui Liu, Pan Lu, Chunyuan Li, Chaowei Xiao, Kai-Wei Chang, Dan Roth, Sheng Zhang, Hoifung Poon, and Muhao Chen. MuirBench: A Comprehensive Benchmark for Robust Multi-image Understanding. *arXiv preprint arXiv:2406.09411*, 2024a. URL https://arxiv.org/abs/2406.09411.

Jiahui Wang, Zuyan Liu, Yongming Rao, and Jiwen Lu. SparseMM: Head sparsity emerges from visual concept responses in MLLMs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025a. URL https://arxiv.org/abs/2506.05344.

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. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024b. doi: 10.48550/arXiv.2409.12191. URL https://arxiv.org/abs/2409.12191.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, Guanzhou Chen, Zichen Ding, Changyao Tian, Zhenyu Wu, Jingjing Xie, Zehao Li, Bowen Yang, Yuchen Duan, Xuehui Wang, Zhi Hou, Haoran Hao, Tianyi Zhang, Songze Li, Xiangyu Zhao, Haodong Duan, Nianchen Deng, Bin Fu, Yinan He, Yi Wang, Conghui He, Botian Shi, Junjun He, Yingtong Xiong, Han Lv, Lijun Wu, Wenqi Shao, Kaipeng Zhang, Huipeng Deng, Biqing Qi, Jiaye Ge, Qipeng Guo, Wenwei Zhang, Songyang Zhang, Maosong Cao, Junyao Lin, Kexian Tang, Jianfei Gao, Haian Huang, Yuzhe Gu, Chengqi Lyu, Huanze Tang, Rui Wang, Haijun Lv, Wanli Ouyang, Limin Wang, Min Dou, Xizhou Zhu, Tong Lu, Dahua Lin, Jifeng Dai, Weijie Su, Bowen Zhou, Kai Chen, Yu Qiao, Wenhai Wang, and Gen Luo. InternVL3.5: Advancing Open-Source Multimodal Models in Versatility, Reasoning, and Efficiency. *arXiv preprint arXiv:2508.18265*, 2025b. URL https://arxiv.org/abs/2508.18265.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, and Dacheng Tao. Divide, Conquer and Combine: A Training-Free Framework for High-Resolution Image Perception

in Multimodal Large Language Models. *arXiv preprint arXiv:2408.15556*, 2024c. URL https://arxiv.org/abs/2408.15556.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. LongVideoBench: A Benchmark for Long-context Interleaved Video-Language Understanding. *arXiv preprint arXiv:2407.15754*, 2024. URL https://arxiv.org/abs/2407.15754.

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and Dahua Lin. PyramidDrop: Accelerating Your Large Vision-Language Models via Pyramid Visual Redundancy Reduction. *arXiv preprint arXiv:2410.17247*, 2024. URL https://arxiv.org/abs/2410.17247.

Guowei Xu, Peng Jin, Ziang Wu, Hao Li, Yibing Song, Lichao Sun, and Li Yuan. LLaVA-CoT: Let Vision Language Models Reason Step-by-Step. *arXiv preprint arXiv:2411.10440*, 2024. URL https://arxiv.org/abs/2411.10440.

Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. XAttention: Block Sparse Attention with Antidiagonal Scoring. *arXiv preprint arXiv:2503.16428*, 2025. URL https://arxiv.org/abs/2503.16428.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. VisionZip: Longer is better but not necessary in vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://arxiv.org/abs/2412.04467.

Huanjin Yao, Jiaxing Huang, Wenhao Wu, Jingyi Zhang, Yibo Wang, Shunyu Liu, Yingjie Wang, Yuxin Song, Haocheng Feng, Li Shen, and Dacheng Tao. Mulberry: Empowering MLLM with o1-like Reasoning and Reflection via Collective Monte Carlo Tree Search. *arXiv preprint arXiv:2412.18319*, 2024. URL https://arxiv.org/abs/2412.18319.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & Verify: Lossless Large Language Model Acceleration via Self-Speculative Decoding. *arXiv preprint arXiv:2309.08168*, 2023. URL https://arxiv.org/abs/2309.08168.

Xuan Zhang, Cunxiao Du, Sicheng Yu, Jiawei Wu, Fengzhuo Zhang, Wei Gao, and Qian Liu. Sparse-to-Dense: A Free Lunch for Lossless Acceleration of Video Understanding in LLMs. *arXiv preprint arXiv:2505.19155*, 2025. URL https://arxiv.org/abs/2505.19155.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. Sparse-VLM: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*, 2024. URL https://arxiv.org/abs/2410.04417.

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Zhengyang Liang, Shitao Xiao, Minghao Qin, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. MLVU: Benchmarking Multi-task Long Video Understanding. *arXiv preprint arXiv:2406.04264*, 2024. URL https://arxiv.org/abs/2406.04264.

A SUPPLEMENTARY MATERIAL

This supplement gives the assumptions and proofs underlying exact verification, the cost and storage analysis, the experimental settings, and the complete reference algorithms for `SpecKV`. It then analyzes the reported ablations, clarifies the accounting behind the runtime breakdown in the main paper, and presents two complementary views of a logged video response. The introductory motivation figure and the main runtime figure are not repeated here.

A.1 THEORETICAL ANALYSIS

A.1.1 LOSSLESS VERIFICATION UNDER ADAPTIVE CONTEXT SELECTION

Context selection changes the proposal distribution, whereas exact verification determines the distribution of the committed output. We formalize this distinction by conditioning on the complete drafting state. The argument specializes standard speculative sampling (Leviathan et al., 2023; Chen et al., 2023) to a context selector that changes after verification; it does not require the selected context to approximate the full context in any particular norm.

Let X denote a committed textual prefix and S the state available before proposing its next token. The state includes the active context, cached draft representations, selection randomness, and feedback from completed verification rounds. Conditioned on (X, S), let $q_S(w)$ be the actual proposal probability and let $p(w) = p(w \mid \mathbf{X}, X)$ be the target conditional with the full multimodal input $\mathbf{X}$. The target conditional is independent of S once its input and prefix are fixed.

Proposition 1 (Exactness with adaptive drafting context)**.** *Suppose that (i) target verification computes the unmodified full-context conditionals, (ii) acceptance and correction use the proposal distributions that actually generated the candidates, and (iii) context updates depend only on information available before the corresponding proposals are drawn. Then exact speculative sampling preserves the target sequence distribution in exact arithmetic, irrespective of the context-selection policy.*

Proof. Fix (X, S). A proposal w with $q_S(w) > 0$ is accepted with probability $\min\{1, p(w)/q_S(w)\}$. Its accepted probability mass is $\min\{p(w), q_S(w)\}$. The total rejection probability is $Z_S = \sum_v [p(v) - q_S(v)]_+$, where $[z]_+ = \max(z, 0)$. For $Z_S > 0$, drawing a correction from the normalized positive residual gives the emitted-token law

$$\begin{aligned} \Pr(W = w \mid X, S) &= \min\{p(w), q_S(w)\} + Z_S \frac{[p(w) - q_S(w)]_+}{Z_S} \\ &= p(w). \end{aligned} \tag{5}$$

If $Z_S = 0$, the two distributions coincide and rejection cannot occur. Tokens outside the proposal support receive their target mass through the correction term. All probabilities refer to the distributions after any agreed decoding transformations, rather than to unprocessed logits. Averaging over S leaves $p(w)$ unchanged. Applying the same identity at each position reached along the accepted prefix, and sampling any full-acceptance bonus directly from the target, establishes the sequence law by induction. Identical EOS and output-cap rules preserve the distribution of the stopped sequence. □

The proposition allows an imperfect or stale selection reference and approximate draft states: these can reduce acceptance but do not change the target conditional. However, recomputing a proposal probability after changing its context does not satisfy assumption (ii). For stochastic decoding, the original proposal distribution, or sufficient state to reconstruct it exactly, is required for residual sampling. The proof concerns sequential speculative sampling and does not, by itself, establish correctness for arbitrary stochastic tree traversal.

A.1.2 GREEDY VERIFICATION AND CANDIDATE TREES

Our reported experiments use greedy decoding. In this setting, losslessness means equality to the target's autoregressive token sequence, under identical input processing, logit processing, tie-breaking, and stopping rules.

Proposition 2 (Greedy output equivalence)**.** *If each accepted candidate equals the target argmax at its committed prefix, and a target-argmax token is emitted whenever no candidate continuation matches, adaptive draft-context selection preserves the target's greedy output sequence.*

Proof. The initial prefixes agree. If the committed prefixes agree at a verification position, full-context verification evaluates the same target conditional as autoregressive decoding. Either a matching candidate or the correction supplies its argmax token, so the extended prefixes also agree. Induction proves equality until EOS or the shared generation cap. □

For a candidate tree, each node must attend only to the committed prefix and its ancestors. Verification follows the target-argmax continuation from the root; sibling branches must not influence one another's states. Nodes on unselected branches are discarded. This extends the greedy argument to trees without treating all verified nodes as accepted output. Identical weights alone are insufficient: the verifier's cache and attention mask must represent the full committed prefix. Finite-precision differences between batched verification and one-token decoding can also alter near-tied argmax decisions; token-level agreement is therefore an implementation check separate from the exact-arithmetic argument.

A.1.3 The Cost–Acceptance Trade-off

The relevant efficiency quantity is time per committed output token, which depends on both proposal acceptance and the cost of producing and verifying proposals. Consider a timed decoding segment with R rounds. Round t commits g_t new tokens and incurs wall time C_t. When execution is sequential, we decompose $C_t = D_t + V_t + S_t + U_t + O_t$ into drafting, verification, selection, cache updates, and residual overhead. For asynchronous execution, the categories must instead partition the measured critical path without double-counting overlapping work. With $N = \sum_t g_t$, $\bar{C} = R^{-1} \sum_t C_t$, and $\bar{g} = R^{-1} \sum_t g_t$, the realized latency is

$$\frac{T_{\text{dec}}}{N} = \frac{\sum_{t=1}^{R} C_t}{\sum_{t=1}^{R} g_t} = \frac{\bar{C}}{\bar{g}}, \qquad \text{Speedup} = \frac{T_{\text{AR}}/N}{\bar{C}/\bar{g}}. \tag{6}$$

This is a finite-run accounting identity, not a ratio of expectations; it requires no independence assumption on consecutive acceptance events. For an ordinary nonterminal chain round, $g_t = a_t + 1$, where a_t counts accepted draft tokens and the extra token is a correction or bonus. EOS, output caps, and root handling can change this relation. We therefore use actual committed counts in Equation 6. The request-averaged τ in the main tables is a separate summary and must not be substituted directly for $\bar{g} - 1$.

Proposition 3 (Condition for a net benefit from adaptation)**.** *For static and dynamic selectors evaluated on the same output token sequence, with positive mean round costs and progress, dynamic selection reduces decoding time if and only if*

$$\frac{\bar{g}_{\text{dyn}}}{\bar{g}_{\text{stat}}} > \frac{\bar{C}_{\text{dyn}}}{\bar{C}_{\text{stat}}}. \tag{7}$$

Proof. Substitute each selector's mean cost and committed count into Equation 6 and cross-multiply positive denominators. □

Writing $\Delta g = \bar{g}_{\text{dyn}} - \bar{g}_{\text{stat}}$ and $\Delta C = \bar{C}_{\text{dyn}} - \bar{C}_{\text{stat}}$ yields $\Delta g/\bar{g}_{\text{stat}} > \Delta C/\bar{C}_{\text{stat}}$. Thus, when adaptation raises round cost, its relative increase in committed tokens must exceed the relative cost increase. Conversely, lower drafting cost can improve speed even without a larger accepted prefix. A fixed active budget does not guarantee equal round costs because selection, cache movement, and verification work can differ.

Including initialization gives $T_{\text{total}} = I + N\bar{C}/\bar{g}$, where I contains visual encoding, prefill, and any other work preceding the timed decoding segment. Consequently, decode-only gains do not imply equal end-to-end gains, particularly for short outputs or expensive candidate-bank construction.

A.1.4 COMPUTATIONAL AND STORAGE COMPLEXITY

Let N_v and N_t denote available visual and textual candidate counts, $N_c = N_v + N_t$, and d the target representation width. Scoring relevance costs $O(N_c d)$; forming the weighted textual reference costs $O(N_t d)$; scoring visual complementarity costs $O(N_v d)$. The total descriptor computation is therefore $O(N_c d)$ per refresh. A single textual reference avoids materializing the $N_v \times N_t$ matrix required by all-pairs cross-modal comparisons. It does not eliminate the need to read the candidate descriptors. Selecting at most B positions can be implemented in $O(N_c \log B)$ time with bounded heaps for $B \geq 2$ (linear selection suffices for $B = 1$); the actual cost of device-side selection depends on its kernel and memory layout.

For a draft decoder with L_q layers, H_q^{KV} KV heads, head width d_h, and b bytes per scalar, define $\kappa_q = 2L_q H_q^{\mathrm{KV}} d_h b$ as the stored KV bytes per candidate position. A full draft-native bank requires approximately $N_c \kappa_q$ bytes, and a separately materialized active buffer adds at most $B\kappa_q$ bytes. Target descriptors require a further $N_c d b_h$ bytes, where b_h is their scalar size. These quantities exclude model weights, the target KV cache, verification workspaces, indices, and allocator overhead. Rebuilding a compact active buffer requires $O(B\kappa_q)$ data movement per refresh.

The active budget bounds historical context reads during drafting; it does not bound total resident cache storage. Newly proposed tree nodes also require temporary KV storage in addition to the selected historical context. Shared model parameters in Self-SD do not imply that approximate draft states can replace the target's full-context cache. Whether implementations can share other buffers is a separate memory optimization and is not assumed in these bounds.

A.2 DETAILED EXPERIMENT SETTINGS

A.2.1 BENCHMARK DETAILS

The main experiments evaluate decoding efficiency on seven benchmarks. The image group covers visual text understanding and reasoning, while the video group covers detailed description and question answering.

Image benchmarks. OCRBench (Liu et al., 2024b) includes text recognition, scene-text and document question answering, information extraction, and handwritten mathematical-expression recognition. MMSpec (Shen et al., 2026) covers six vision-language speculative-decoding categories: general and text-based visual question answering, captioning, chart understanding, complex reasoning, and multi-turn conversation. MathVista (Lu et al., 2024) combines visual interpretation with mathematical inference over diagrams, charts, and images. ScienceQA (Lu et al., 2022) uses multimodal multiple-choice science questions with supporting textual or visual context.

Video benchmarks. VDC (Chai et al., 2024) evaluates detailed captions involving subjects, backgrounds, fine-grained details, and camera motion. VDD (LMMs-Lab, 2024) uses holistic descriptions of scenes, entities, actions, and their temporal development. MovieChat (Song et al., 2023) evaluates open-ended video question answering. These task descriptions explain the evaluation scope; they do not specify the number or identity of requests used in a run.

A.2.2 BASELINE DETAILS

The five baselines in Tables 1 and 2 retain the full target context: **Vanilla** performs target autoregressive decoding; **SD-Tree** uses uncompressed historical context for tree-based drafting; **SD-Rand** randomly selects visual context for drafting; **StD** (Zhang et al., 2025) uses sparse top-K draft attention with dense verification; and **SpecVLM** (Ji et al., 2025) combines verifier-guided visual retention with spatially uniform pruning. Their active-context policies differ, so a shared method label or nominal visual budget alone does not establish an equal-total-context comparison. The full-context SD control in Table 3 is kept separate from the main tables' SD-Tree results; the manuscript does not provide sufficient execution detail to identify the two configurations.

A.2.3 MODEL DETAILS

Standard-SD uses Qwen2.5-VL-32B/7B (Bai et al., 2025) and LLaVA-OneVision-72B/7B (Li et al., 2024a) as target/drafter pairs, following Section 4.1. Self-SD uses the corresponding 7B model for

both roles, sharing parameters while maintaining full-context verification and restricted drafting. No additional training is introduced. Target and draft must use compatible text-token identities, and differing visual grids require the alignment described in Section A.3.3.

A.2.4 Implementation and Evaluation Protocol

The configuration below follows the experimental setting in Section 4.1. The historical budget and the output limit are separate quantities: B bounds selected cached history, whereas $L_{\max}$ bounds newly generated output tokens.

Table 4: **Reported evaluation configuration.** Output caps are shared by methods within a benchmark; they do not set the historical KV budget.

Item	Setting
Decoding	Greedy, with no minimum output length
Stopping	EOS or $L_{\max} = 2048$ new tokens (images), 512 (videos)
Historical draft budget	$B = 512$, except in the budget sweep
Visual mixture	$\lambda = 0.25$ in the experimental setting; varied explicitly in the sweep
Candidate trees	Fixed trees, maximum drafting depth five
Hardware	Two NVIDIA H200 GPUs
Training	No additional model training
Target verification	Complete multimodal context

The maximum tree depth specifies the longest candidate path, not the total number of verified nodes or the accepted-length count. The same target input, decoding rule, and output cap define the comparison on each request. The historical budget excludes temporary candidate-tree storage, as detailed in Section A.3.

Timing interval. Timing starts after prefilling and generation of the first target token. It includes all subsequent drafting, context selection, target verification, and cache updates through termination. Visual encoding and prefilling are excluded. Work to consume the first emitted token or to update its caches belongs inside the timed segment if it occurs after the timer starts. Thus, a time fraction that includes prefilling and a decode-only time fraction have different denominators and are not interchangeable.

Metric aggregation. For benchmark requests $j = 1, \dots, m$, let $n_j^{(a)}$ be the number of timed output tokens under method a, $T_j^{(a)}$ its decoding duration, and $k_{jt}^{(a)}$ the number of accepted draft tokens in round t. With $R_j^{(a)}$ verification rounds, the reported conventions are

$$v_a = \frac{\sum_j n_j^{(a)}}{\sum_j T_j^{(a)}}, \qquad S_a = \frac{\sum_j T_j^{(\mathrm{AR})}}{\sum_j T_j^{(a)}},$$

$$\tau_a = \frac{1}{m} \sum_{j=1}^{m} \frac{1}{R_j^{(a)}} \sum_{t=1}^{R_j^{(a)}} k_{jt}^{(a)}.$$

The accepted count excludes root, correction, and bonus tokens, and discards branches or suffixes not committed to the output. Requests with no verification round have no defined per-request accepted-length mean; they must be identified separately rather than introducing a division by zero. The reported summaries do not enumerate such cases.

For identical timed output counts, $S_a = v_a / v_{\mathrm{AR}}$. This ratio of total times differs from the mean of per-request speedups. Likewise, the request-averaged τ_a differs from pooling all rounds, so it cannot replace $\bar{g} - 1$ in Equation 6. The "Avg." column is the arithmetic mean of benchmark-level speedups, not a pooled throughput across tasks. Standard-SD and Self-SD use the target checkpoint in their own setting as the AR reference; ratios for different target sizes are not measurements of acceleration of one identical model.

Reproduction scope. The configuration and aggregation rules above define the reported comparison. The accompanying manuscript source does not contain a complete, run-linked record of evaluated request IDs, input preprocessing, prompt templates, checkpoint and software revisions, precision, device placement, batching, or the full tree topology. It also does not include paired token-output logs or repeated timing records. Accordingly, we do not infer dataset coverage, output-agreement rates, confidence intervals, or backend-specific behavior from the aggregate tables. These execution details are distinct from the mathematical reference procedure in Section A.3.

A.3 Algorithms and Cache Implementation

Algorithms 1 and 2 specify the greedy decoding loop and text-prioritized selection in Section 3. The target always verifies with its complete context. The algorithms define a reference procedure, including explicit boundary conventions. As discussed in Section A.1.1, exactness additionally requires the inference backend to preserve the specified target context, attention masks, and verification rule.

Table 5: **Cache roles and verification invariants.** The historical budget B applies to the active draft view, not to every stored tensor. These are logical roles; sharing physical storage is valid only when it preserves the corresponding states and visibility.

Object	Role and required invariant
Target cache	Full-context states for the committed prefix; approximate draft states never replace verifier states.
Draft candidate bank	Materialized draft-native historical KV, including inactive entries available for reselection.
Active draft view	At most B historical positions gathered from the bank, with original position metadata preserved.
Temporary path KV	Pending committed tokens and proposed nodes outside the selected history; tree branches see only their ancestors.
Target descriptors	Same-layer target-space vectors used to score positions, not substitutes for draft-native keys or values.

A.3.1 Complete SpecKV Decoding Algorithm

Let $\mathcal{B}_q$ contain materialized draft-native KV entries, indexed by original position, and let $\mathcal{H}_p$ contain aligned target descriptors. The selected historical view remains fixed within a drafting round. Proposed nodes use temporary branch-specific states; only committed, materialized positions subsequently enter the bank. A correction or bonus can be committed before its own KV is computed, so the algorithm explicitly tracks pending cache work for such tokens.

The state that predicts a token is not the state after consuming that token. Consequently, emitted corrections and bonuses are not treated as historical candidates until their KV and descriptors exist. The first target token is outside the main tables' decoding timer, while subsequent work to consume it and update the caches belongs inside that timer, as specified in Section A.2.4.

A.3.2 Verification-Guided Context Selection

Selection compares candidate descriptors with g in the same target representation space. Descriptors are used only for scoring and are never substituted for the corresponding draft keys or values. Algorithm 2 implements the allocation in Equation 4: text receives up to B slots, and visual candidates fill the remaining allowance. No additional text-quota parameter or global cross-modal score calibration is required.

Min–max normalization sends equal scores to zero, so uniform aggregation must be an explicit branch rather than a consequence of Equation 2. The reference specification uses uniform text weights when their sum is zero, and uses relevance-only visual ranking when text is absent or its aggregate has zero norm. A zero-norm descriptor is assigned zero cosine in this specification; non-finite inputs require an error rather than an arbitrary ranking. Equal selection scores are resolved by original position. These conventions make the reference selection deterministic without altering the full-context verification rule.

Algorithm 1 `SpecKV`: greedy decoding with recoverable historical context

Require: Target $\mathcal{M}_p$, drafter $\mathcal{M}_q$, input $\mathbf{X}$, budget $B \geq 0$, tree $\mathcal{D}$, weight λ, positive output cap

Prefill the target with full $\mathbf{X}$ and initialize its complete cache
Prefill the drafter; construct $\mathcal{B}_q$ and aligned $\mathcal{H}_p$
$g \leftarrow$ target state at the final prompt position
$Y \leftarrow$ first target-greedy token; record its pending cache work
while Y contains no EOS and the output cap has not been reached **do**
    $\mathcal{C}_q \leftarrow \textsc{SelectContext}(\mathcal{B}_q, \mathcal{H}_p, g, B, \lambda)$
    Consume pending committed tokens; draft with $\mathcal{C}_q$ and $\mathcal{D}$
    Verify with full target history and branch-causal attention masks
    Obtain the target-matching path A, truncated at EOS and the output cap
    if $|A| > 0$ **then**
        $g \leftarrow$ target state after consuming the last token in A
    end if
    Append A to Y
    if generation has not terminated **then**
        Append the target-greedy correction or full-acceptance bonus to Y
    end if
    Commit valid path states; discard rejected nodes and unused branches
    Append newly materialized committed entries to $\mathcal{B}_q, \mathcal{H}_p$
    Keep any unmaterialized committed token pending for the next round
end while
return Y

Algorithm 2 Text-prioritized selection: reference specification

function $\textsc{SelectContext}(\mathcal{B}_q, \mathcal{H}_p, g, B, \lambda)$
    $\mathcal{V}, \mathcal{T} \leftarrow$ materialized visual and textual candidate positions
    Compute $r_i = \cos(g, h_i)$; normalize nonempty modalities separately
    $b_T \leftarrow \min\{|\mathcal{T}|, B\}$
    $\mathcal{I}_T \leftarrow$ the b_T highest-relevance textual positions
    $b_V \leftarrow \min\{B - b_T, |\mathcal{V}|\}$
    $s_i \leftarrow \tilde{r}_i$ for $i \in \mathcal{V}$ ▷ relevance fallback
    if $b_V > 0$ and $|\mathcal{T}| > 0$ **then**
        $w_j \leftarrow \tilde{r}_j$ for $j \in \mathcal{T}$
        if $\sum_{j \in \mathcal{T}} w_j = 0$ **then**
            $w_j \leftarrow 1$ for $j \in \mathcal{T}$ ▷ explicit uniform fallback
        end if
        $\bar{h} \leftarrow \sum_{j \in \mathcal{T}} w_j h_j / \sum_{j \in \mathcal{T}} w_j$
        if $\bar{h}$ has a nonzero norm **then**
            $u_i \leftarrow 1 - \cos(h_i, \bar{h})$; normalize u within $\mathcal{V}$
            $s_i \leftarrow \lambda \tilde{r}_i + (1 - \lambda)\tilde{u}_i$
        end if
    end if
    $\mathcal{I}_V \leftarrow$ the b_V highest-scoring visual positions
    $\mathcal{I} \leftarrow \mathcal{I}_T \cup \mathcal{I}_V$, sorted by original logical position
    return draft-native KV at $\mathcal{I}$ with original position metadata
end function

A.3.3 Cache Updates, Position Handling, and Rollback

Changing the active view does not delete inactive bank entries or recompute their representations. In particular, a committed textual entry retains the draft state produced under its original context. Reusing that entry after refresh need not equal recomputing it with the new view: it defines a proposal process still covered by Section A.1.1, provided that verification uses the actual proposal distribution and the unmodified target context.

Gathered entries retain their original logical positions and model-specific spatial or temporal metadata. Their physical buffer indices do not replace those positions. For matching visual grids, the descriptor association follows the aligned visual positions. For different grids, the coordinate-overlap weighting stated in Section 3.3 can be written as

$$h_i^{\text{aligned}} = \frac{\sum_j a_{ij} h_j^p}{\sum_j a_{ij}}, \qquad a_{ij} = \mu(R_i^q \cap R_j^p), \tag{8}$$

where the footprints R_i^q and R_j^p must be expressed in common image or spatiotemporal coordinates, and μ measures their overlap. This expression formalizes the overlap-weighted association in the main method. An eligible mapping requires a positive denominator and compatible crop, merging, and temporal metadata; it cannot be obtained by comparing descriptors from unrelated model spaces.

The overlap weights are applied to the same-layer target descriptors; cosine normalization is then applied to the resulting aligned vector. The relevance-weighted textual mean likewise uses the descriptors specified in Equation 2. Keeping unit-normalized copies for cosine scoring does not redefine that mean as an average of normalized descriptors. The selected logical positions are gathered across the draft layers; the rule is not a separate per-head or per-layer budget. Draft and target tokens must also share compatible text-token identities for direct verification; same-family names alone do not establish this compatibility.

After verification, only valid committed path states enter the target cache. Rejected suffixes and unused branches are removed, and pending tokens are consumed before being used as cached history. In Self-SD, parameter sharing does not permit copying approximate draft KV into the full-context target cache. Temporary candidate-tree storage, including a cached root when present, is separate from the selected historical budget B.

A.3.4 Initialization and Boundary Cases

The initial reference is the target state at the final prompt position. When a round accepts draft tokens, the last accepted token supplies the next reference; otherwise, the previous reference is retained. A rejected state is never used as accepted progress. Full acceptance produces a target bonus only if generation has not already ended. Tokens after EOS or beyond the generation cap are neither committed nor included in the accepted-length count.

If $|\mathcal{T}| \le B$, all available text fits and the remaining slots are allocated to vision. When $|\mathcal{T}| \ge B$, the text allowance reaches B and $b_V = 0$; text selection then ranks a bounded subset when necessary. A bank smaller than B leaves unused capacity, and $B = 0$ yields no selected historical entries. Any protected structural entries must be included in the same accounting rather than added outside B. The video case in Section A.6 illustrates visual-budget saturation; it does not establish the saturation frequency across benchmarks.

Consistency of the textual reference. The allocation also implies

$$b_V > 0 \implies |\mathcal{T}| < B \implies \mathcal{I}_T = \mathcal{T}.$$

Thus, whenever visual candidates can enter the active view, the textual reference aggregates retained text. When text exceeds the budget, $b_V = 0$ and the reference cannot affect visual admission. This follows from the text-prioritized rule, not from an assumption that a textual summary perfectly replaces the visual evidence it describes.

A.3.5 Reference-Contract Checks

The following invariants make the verification and cache specification checkable independently of aggregate speed. They are requirements of the reference procedure, not reported agreement rates of a GPU implementation.

Table 6: **Boundary and consistency checks for the reference procedure.** The expected outcomes follow from the stated algorithm and verification rule.

Condition	Required outcome
Empty modality / short bank	Normalize only nonempty sets; select no more than the available positions or B.
Equal textual relevance	Use an explicit uniform reference, avoiding a zero denominator.
No valid text reference	Rank visual candidates by relevance alone.
Zero accepted draft tokens	Retain g; commit only the target continuation and track pending states.
EOS / cap on a path	Do not commit, cache, or count any suffix beyond termination.
Sibling candidate branches	No cross-branch attention; discarded nodes never enter the target history.
Rebuilt active view	Preserve logical positions and leave inactive bank entries recoverable.
Text saturates B	Allocate no historical visual slots; do not treat this as deleting the visual bank.

For a finite-precision backend, a separate matched-AR check must compare committed token sequences under the same preprocessing, tokenizer, logit transformations, tie-breaking, and stopping configuration. Comparing only task scores or total accepted length cannot detect every token mismatch. The supplied aggregate measurements do not give the number of such checked requests or their disagreement rate, so none is inferred here.

A.4 Extended Ablation Studies

We analyze the comparisons in Table 3 and Figure 3 for Qwen2.5-VL-32B/7B under Standard-SD. Compression scope and refresh policy cover VDC, VDD, MathVista, and OCRBench; the scoring and budget sweeps concern VDC. Relative throughput changes are calculated from the reported table, not additional benchmark runs. The plotted sweeps are single-run measurements, so small differences are interpreted descriptively rather than as evidence of statistical significance.

A.4.1 Compression Scope

Visual-only and text-only compression leave the other modality uncompressed. They therefore test the cost of retaining one modality in full, rather than compare allocations at an equal total budget. Visual-only compression is faster on all four tasks: its speedups are $2.53\times$, $1.88\times$, $2.17\times$, and $1.68\times$ on VDC, VDD, MathVista, and OCRBench, compared with $1.60\times$, $1.22\times$, $1.39\times$, and $1.10\times$ for text-only compression. Restricting visual access accounts for the larger efficiency gain in this comparison.

Full-context SD attains the largest τ in each column but is slower than visual-only compression and both shared-budget selectors. On VDC, moving from full-context SD to dynamic `SpecKV` reduces τ from 3.55 to 3.24, while increasing speedup from $1.64\times$ to $2.73\times$. Thus, maximizing accepted length alone does not minimize decoding time, as Equation 6 shows. This within-table comparison does not equate the ablation's full-context control with the separately reported SD-Tree baseline.

A.4.2 Static versus Dynamic Selection

The update comparison uses $B = 512$ and text-prioritized allocation. Static selection is the initialization-based control, whereas dynamic selection refreshes its criterion from verification feedback. Both remain subject to the same total historical budget as text grows; an initialization-based control must still account for newly committed text. The aggregate table does not identify the exact overflow implementation for that control, so it cannot establish equivalence of alternative overflow policies. Table 7 expands this comparison into accepted lengths and relative throughput gains, keeping each benchmark paired.

Dynamic refresh increases throughput by 4.58%–7.22% and τ by 0.04–0.05 across the four tasks. Their shared direction supports refreshing the selector beyond its initial estimate. The throughput gain should not, however, be attributed solely to acceptance: cache movement and work per round may also change, and the request-averaged τ is distinct from the committed-token quantity $\bar{g}$ in Equation 6.

Table 7: **Benefit of dynamic refresh in the four-task ablation.** Throughputs and τ reproduce Table 3. Gain is $100(v_{\mathrm{dyn}}/v_{\mathrm{stat}} - 1)$, computed from rounded throughputs, rather than a percentage-point change in AR speedup.

	Tokens/s		τ		Throughput
Benchmark	Static	Dynamic	Static	Dynamic	gain (%)
VDC	20.75	21.70	3.20	3.24	4.58
VDD	19.54	20.95	3.20	3.24	7.22
MathVista	25.83	27.30	4.06	4.11	5.69
OCRBench	19.17	20.31	3.05	3.09	5.95

A.4.3 Relevance and Complementarity

The weight sweep varies the visual score $s_i = \lambda \tilde{r}_i + (1 - \lambda)\tilde{u}_i$ for $\lambda \in \{0, 0.25, 0.5, 0.75, 1\}$ at $B = 512$. At $\lambda = 1$, visual ranking uses relevance alone. At $\lambda = 0$, visual relevance is removed from the final score, but textual relevance still weights the reference in Equation 2. Thus, complementarity-only visual ranking is not an ablation of all target-derived guidance; it retains a target-conditioned textual reference.

The experimental configuration in Section 4.1 specifies $\lambda = 0.25$. The endpoint definitions explain which term changes; they do not imply that one mixture is universally optimal. In particular, accepted length alone does not identify the fastest mixture, because scoring, drafting, and verification costs all affect latency. Changing the mixture under a fixed refresh policy also does not independently separate interactions between refresh and complementarity.

A.4.4 Alternative Relevance Signals

The relevance comparison disables visual complementarity and considers random ranking, draft attention, target attention, and target hidden-state cosine. These variants distinguish the cost reduction from a smaller draft context from the choice of relevance signal. Attention-based alternatives additionally depend on the queried positions, layers, head aggregation, and attention-extraction path. Their comparison must include that extraction cost in the same decoding interval; it is not a comparison of similarity formulas in isolation.

The weight and signal comparisons answer different questions: the former mixes two scoring terms, whereas the latter compares relevance sources. They cannot be combined into an extra controlled ablation without matching their execution configurations. We introduce no additional signal measurements or statistical ranking here. The numerical cross-task reanalysis in Table 7 is derived solely from the reported scope/update table, not from transformed plotting values.

A.4.5 Budget Sensitivity and Allocation Scope

Section 4.3 considers $B \in \{159, 318, 635, 1269, 2537\}$ on VDC, in addition to the default absolute budget $B = 512$. A budget is held fixed during a response rather than increased as generated text accumulates. The default is not a fixed percentage of every request's visual input or an output-length limit. More context can increase both useful progress and per-round work, so speed need not increase monotonically with B. The cost identity in Equation 6 explains this trade-off without assuming that a larger budget always accelerates decoding.

These experiments distinguish compression scope, refresh, scoring, and total budget. They do not independently determine the best text–vision split or compare relevance-weighted and uniform textual references. The four-task refresh findings therefore have a broader task scope than the VDC-only scoring and budget findings. Section A.6 adds a concrete allocation trajectory, including the boundary at which the active visual allowance vanishes.

A.5 Runtime Accounting and Storage Analysis

The main paper's Figure 4 gives the Qwen2.5-VL-32B/7B runtime breakdown on VDC under Standard-SD. Rather than repeat it, we explain its normalization, the distinction between stage costs and incremental overhead, and the storage implications of recoverability.

A.5.1 Per-Token Accounting and the Static–Dynamic Comparison

Let $W_{a,k}$ be the time assigned to stage k of method a and N_a the committed output-token count in the same timed requests. Then

$$c_{a,k} = 1000 W_{a,k}/N_a, \qquad \ell_a = \sum_k c_{a,k} = 1000/v_a \quad \text{(ms/output token)}.$$

The denominator counts emitted output, not all verified candidate nodes. For overlapping execution, a wall-time stack must partition the critical path rather than double-count concurrent kernel durations.

The main figure reports total latencies of 125.79, 76.92, 48.19, and 46.08 ms/token for AR, full-context SD, static `SpecKV`, and dynamic `SpecKV`. These match, to rounding, the reciprocals of 7.95, 13.00, 20.75, and 21.70 tokens/s in Table 3. This is an accounting check, not an independent experiment. In particular,

$$\text{Latency reduction} = 100(1 - 46.08/48.19) = 4.38\%,$$
$$\text{Throughput gain} = 100(21.70/20.75 - 1) = 4.58\%.$$

The percentages differ because latency and throughput are reciprocals; neither is a percentage-point change in speedup over AR.

The reported drafting costs decrease from 21.0 to 18.1 ms/token, a 2.9 ms/token saving, whereas total latency decreases by 2.11 ms/token. The remaining stages therefore sum to 27.19 and 27.98 ms/token: their combined cost increases by 0.79 ms/token, using the rounded figure values. This explains why the drafting saving is not the total saving. The remainder includes verification and other work, so its increase is not a measurement of selection overhead alone.

A.5.2 Stage Shares versus Incremental Overhead

Reference aggregation, relevance/complementarity scoring, and index selection form one category; bank maintenance and active-KV gathering or repacking form another. A selection-only share excludes cache movement. Conversely, a maintenance stage need not be entirely additional work, since the static baseline may also perform part of it. Incremental overhead is the difference in corresponding costs under matched outputs and timing boundaries, not a stage's fraction of one method's own runtime. The full comparison determines net benefit, as Equation 7 formalizes.

A.5.3 Initialization and End-to-End Latency

The decode-only timer excludes prefill and the first target token (Section A.2.4). If I_a contains all work before that boundary, then for identical timed output count N,

$$T_{a,\text{total}} = I_a + N\ell_a/1000, \qquad S_{a,\text{total}} = \frac{I_{\text{AR}} + N\ell_{\text{AR}}/1000}{I_a + N\ell_a/1000}.$$

Initialization can differ across methods, particularly when a separate draft bank is prefetched. Figure 4 does not supply these initialization terms. We therefore do not infer a numerical end-to-end speedup or combine its per-token costs with an unlinked profile that includes prefilling.

A.5.4 Resident Storage versus Active Access

As derived in Section A.1.4, the draft bank requires about $N_c\kappa_q$ bytes; a separate compact active buffer adds at most $B\kappa_q$, and target descriptors add $N_c d b_h$. Storing both raw and unit-normalized descriptors requires counting both copies. A view that aliases bank storage can avoid a duplicate active buffer, but not the bank itself. These quantities exclude weights, the target cache, temporary tree states, indices, and allocator overhead.

Recoverability trades persistent storage for flexible sparse access. A smaller B reduces selected historical reads, not necessarily peak resident memory. In Self-SD, weights can be counted once, but approximate draft states cannot replace full-context target states. These expressions are analytical storage accounting, not measured peak-memory results. Neither the runtime comparison nor the static/dynamic ablation establishes a universal overhead fraction or an optimal intermediate refresh schedule.

A.6 Visualizations of Context Adaptation

The selection-overlap diagnostic and redundancy schematic appear in Figure 1 of the main paper. Here we examine the logged VDC doc16 response through two complementary views: which visual positions enter the active context, and how their available capacity changes as text accumulates. Both figures describe the same Qwen2.5-VL-32B/7B run at $B = 512$; they are separate from the introductory diagnostic.

A.6.1 Spatial Selection in a Logged Video Response

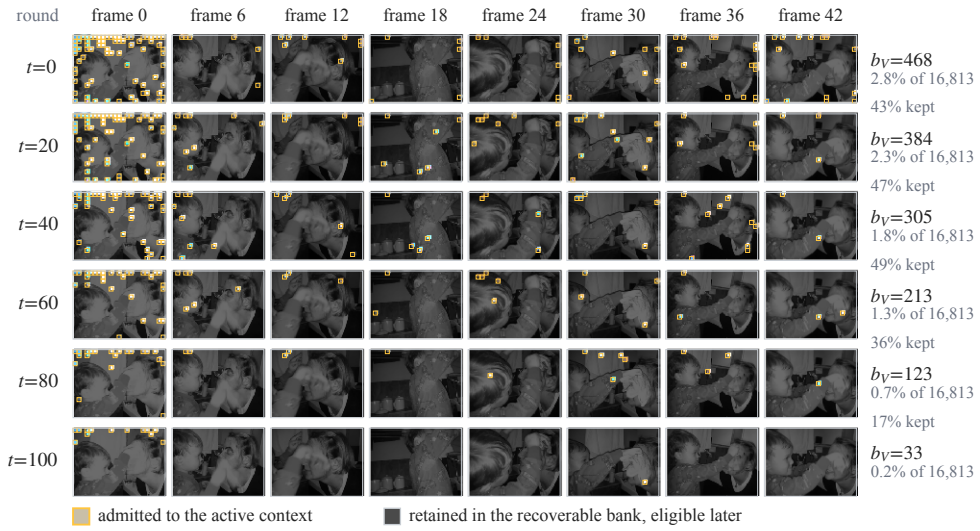

Figure 5: **Spatially nonuniform selection across verification checkpoints.** Qwen2.5-VL-32B/7B on VDC doc16 at $B = 512$. Eight fixed temporal positions from a 43-position video are shown. Bright patches are admitted to the active context; dark patches remain in the recoverable bank. Rows correspond to $t = 0, 20, 40, 60, 80, 100$, not consecutive individual rounds. The right-hand counts cover the entire video, not only the eight displayed columns.

The clip supplies 16,813 visual KV entries. At the displayed checkpoints, b_V is 468, 384, 305, 213, 123, and 33, respectively, corresponding to a decrease from approximately 2.8% to 0.2% of the visual bank. Selection is concentrated within and across frames rather than spatially uniform. The displayed maps describe where access is concentrated, not which frame is causally necessary.

The annotated retention rates are 43%, 47%, 49%, 36%, and 17% between successive displayed checkpoints. Each comparison spans 20 verification rounds. Because the visual allowance also shrinks, these rates reflect both changing selections and fewer available visual slots; they do not isolate evidence-demand drift. Darkening a patch denotes inactive access rather than deleting its cached representation, consistent with the recoverable-bank design.

A.6.2 Text Growth and Visual-Budget Saturation

Under Equation 4, admitting more text leaves fewer slots for direct visual access. At the six checkpoints in Figure 5, $512 - b_V$ gives textual allocations of 44, 128, 207, 299, 389, and 479. These counts follow from the allocation rule and displayed visual counts; they are not independently measured tokenized-prefix lengths. With a fixed budget and nondecreasing materialized text, the visual allowance is nonincreasing, even when relevance changes across rounds.

The final 44 tokens in Figure 6 illustrate the saturation boundary: the active historical view can become entirely textual. This does not establish that visual evidence is unnecessary. Cached textual representations may already contain visual information, and the target still verifies with its complete multimodal context. Inactive visual entries remain stored, but refreshing their scores cannot readmit

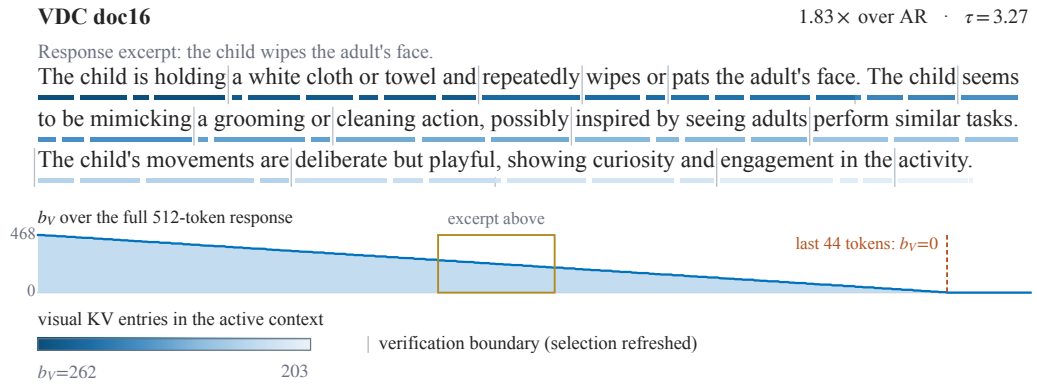

Figure 6: **Visual allocation over the same VDC response.** Token underlines encode the visual budget at the preceding selection checkpoint, and vertical rules mark verification boundaries. They are not token-importance, correctness, or acceptance labels. The excerpt spans $b_V = 262 \rightarrow 203$; the lower curve covers all 512 generated tokens, with $b_V = 0$ for the final 44. The $1.83\times$ paired-AR speedup and $\tau = 3.27$ summarize this case rather than benchmark-wide performance.

them while $b_V = 0$. Under unchanged B and continued text growth, the allocation does not reopen visual slots.

A.6.3 Interpretation and Scope

Figure 5 shows where the drafter reads visual history; Figure 6 shows how much visual history the budget permits. Together they make the distinction between ranking, allocation, and recoverable storage explicit. The single-case timing and accepted length are not estimates of typical request performance, and neither view measures the functional interchangeability of corresponding visual and textual evidence.

A.7 Scope of Cross-Setting Comparisons

The main comparisons cover two model families, Standard-SD and Self-SD, and seven image/video benchmarks. The four-task ablation addresses whether context should be refreshed, while the VDC sweeps address scoring and budget choices. The main runtime breakdown and doc16 visualizations describe execution costs and selection behavior at their respective reporting scopes. These sources provide complementary evidence rather than additional replications of one identical experimental setting.

Target-relative speedup. Standard-SD and Self-SD use different target sizes and therefore different AR references. Their ratios do not measure acceleration of the same target checkpoint. Image and video averages also summarize different task collections, not matched input-length groups. A shared $B = 512$ default standardizes the historical allowance without making the active-context fraction, resident storage, or best budget identical.

Design-specific conclusions. Dynamic refresh improves reported throughput on all four ablation tasks, but scoring sensitivity and budget sensitivity are characterized only on VDC. The qualitative case illustrates one complete allocation trajectory, not the prevalence of that trajectory across requests. Single-run summaries do not establish statistical superiority or variation across requests. In particular, neither target-relative aggregate speedup nor a high mean accepted length ensures that every request improves.

Availability of statistical and output checks. The manuscript source supplies aggregate results and visualization assets, not paired request-level token sequences or timing repetitions. Thus, the correctness proofs and reference-contract checks are not empirical output agreement measurements. A confidence interval would require resampling actual paired requests (whole sessions for multi-turn

data), with timing repetitions distinguished from independent requests. No such interval or failure-rate estimate is derived from rounded table entries here.

A.8 Limitations and Boundary Cases

Similarity is a selection proxy. Hidden-state cosine measures representational alignment, not the causal contribution of a KV entry to acceptance. A single textual aggregate can mix unrelated facts or obscure a fine-grained visual detail. Although the text-prioritized rule ensures that the reference summarizes retained text whenever visual slots remain (Section A.3.4), it does not guarantee semantic coverage or functional interchangeability across modalities.

Feedback can lag behind future requirements. The latest accepted state reflects verified progress rather than a known future evidence requirement. Topic changes and transitions to fine-grained reading can make it less informative. Retaining the previous reference after zero acceptance avoids using a rejected continuation but can leave selection stale. Full-context verification preserves correctness under the stated assumptions, not a long accepted prefix or low decoding latency for every request.

Text priority can exhaust visual access. The VDC case reaches $b_V = 0$ for its final 44 tokens. Inactive visual entries remain stored, but cannot re-enter the active view while text occupies all B slots. With fixed B and continued text growth, refreshing scores alone cannot change this allocation. The case exposes a budget boundary rather than showing that visual grounding is unnecessary or that another allocation would be slower.

Recoverability preserves storage costs. The candidate bank retains inactive entries, trading storage for flexible access. The runtime breakdown shows a net saving in its reported comparison, but does not establish a universal overhead fraction or a peak-memory reduction. Short inputs or outputs, different kernels, and candidate-tree costs can change the balance between saved reads and additional work.

Exactness and empirical coverage are distinct. The proofs assume full-context verification and, for stochastic sampling, the actual proposal probabilities. They do not certify an arbitrary stochastic tree sampler or numerical behavior of a backend. The empirical scope remains the reported greedy experiments and single-run sweeps; aggregate timing and qualitative examples do not establish a token-level agreement rate, repeat-run variability, or performance on untested cache architectures and applications.