Instructions for *ACL Proceedings

Anonymous ACL submission

Abstract

Self-speculative decoding accelerates autoregressive generation by drafting candidate tokens from the verifier's own hidden states. For vision-language models (VLMs), recent VLM-SD methods often make the drafter vision-aware through image-token attention, multimodal drafter adaptation, or cross-modal fusion. We propose **V5e-0**, a minimalist SSD framework whose drafter uses only the verifier's text-side hidden states, with no vision encoder, image-token input, visual cross-attention, or verifier update. V5e-0 reuses the verifier's greedy argmax as a parameter-free root token (*Free-Root*) and uses two lightweight continuation heads to build a single-root depth-3 speculative tree verified in one forward pass. Across fifteen VLMs, V5e-0 achieves a mean wall-clock speedup of $1.89\times$ over autoregressive decoding. Controlled evaluations show a domain-dependent trade-off: V5e-0 trails ViSpec by $0.31\times$ on TextVQA, but preserves AR task accuracy on OCRBench while achieving a $1.46\times$ speedup. Ablations support Free-Root, sequential continuation modeling, and the static depth-3 tree, while tested vision-token variants do not improve the lightweight drafter in our setting. We provide the anonymized implementation, interactive demo, and measurement artifacts here.

1 Introduction

Vision-Language Models (VLMs), such as BLIP-2 (Li et al., 2023), LLaVA (Liu et al., 2023), and Qwen2-VL (Wang et al., 2024b), generate responses from both visual and textual inputs. However, autoregressive VLM inference remains expensive: each generated token requires a verifier forward pass, and visual-token prefixes increase memory traffic and decoder-side computation.

Speculative decoding reduces autoregressive latency by drafting multiple candidate tokens and verifying them with the target model (Stern et al., 2018; Leviathan et al., 2023; Chen et al., 2023). Self-speculative decoding avoids a separate draft model by deriving candidates from the verifier itself or from lightweight auxiliary heads (Zhang et al., 2024a; Cai et al., 2024; Ankner et al., 2024; Li et al., 2024b,a, 2025). When applied to VLMs, recent methods often make the drafter vision-aware: ViSpec (Kang et al., 2025) uses image-token attention, MASSV (Ganesan et al., 2025) adapts a multimodal drafter, and DREAM (Hu et al., 2025) fuses multimodal features into the drafter.

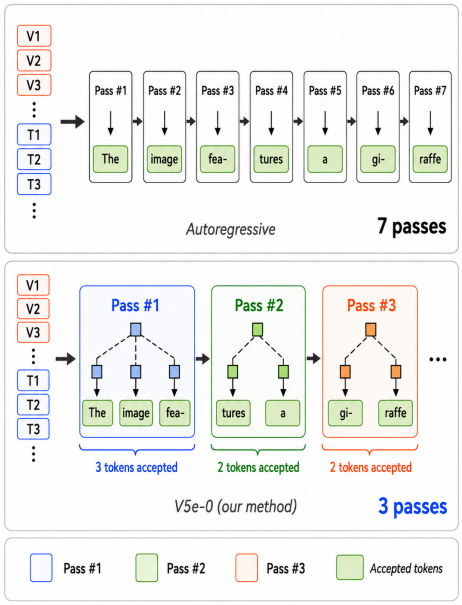

Figure 1: Autoregressive decoding versus V5e-0. Both methods condition on the same input prefix. AR emits one token per verifier pass, whereas V5e-0 can accept multiple greedy-matched tokens per pass through speculative verification.

This paper revisits whether explicit visual processing is required inside the drafter. Our premise is that the verifier has already processed both image and text tokens before producing its language-side hidden state. Thus, this hidden state may already contain the multimodal context needed for specu-

lative continuation. We ask whether VLM speculative drafting can be treated as language-side continuation modeling over verifier hidden states, rather than as an additional multimodal drafting problem.

We propose **V5e-0**, a minimalist self-speculative decoding framework for VLMs. V5e-0 keeps the verifier frozen and uses only the verifier's text-side hidden state to generate draft candidates. It has no vision encoder, no image-token input, and no visual cross-attention. It reuses the verifier's greedy next-token prediction as a parameter-free root token, called *Free-Root*, and uses two lightweight linear continuation heads to construct a single-root depth-3 speculative tree. Figure 1 illustrates the decoding-level effect.

Across fifteen VLMs, V5e-0 achieves a mean wall-clock speedup of $1.89\times$ over autoregressive decoding. Controlled evaluations show a domain-dependent trade-off: ViSpec is faster on TextVQA, while V5e-0 preserves answer-level accuracy on vision-grounded benchmarks with reduced but non-trivial speedup. These results position V5e-0 as a strong minimalist baseline for greedy VLM speculative decoding, not as a universal replacement for vision-aware drafters.

Our contributions are:

- We propose V5e-0, a VLM self-speculative decoding framework whose drafter uses only text-side verifier hidden states, without image-token input or visual cross-attention.
- We introduce Free-Root and a single-root depth-3 speculative tree built from two lightweight continuation heads.
- We evaluate V5e-0 across fifteen VLMs and provide controlled analyses on vision-aware baselines, vision-grounded tasks, design ablations, and output fidelity.

2 Motivation

Recent VLM speculative decoding methods often make the drafter vision-aware. ViSpec (Kang et al., 2025) uses image-token attention, MASSV (Ganesan et al., 2025) adapts a multimodal drafter, DREAM (Hu et al., 2025) fuses multimodal features into the drafting path, and SpecVLM (Huang et al., 2025) uses a vision-aware draft head. These methods differ in implementation, but they share the same design choice: visual information is explicitly routed into the drafter. In contrast, LLM-side self-speculative methods such as Medusa (Cai et al., 2024), Hydra (Ankner et al., 2024), and EAGLE (Li et al., 2024b,a, 2025) show that verifier hidden states can provide useful drafting signals without a separate full draft model. Adjacent parallel-decoding paradigms further reduce the auxiliary-model dependency: CLLMs (Kou et al., 2024) train a single model to be consistent under multi-token jacobi updates, and Kangaroo (Liu et al., 2024a) treats the verifier's own shallow layers as a lossless self-speculative drafter. We focus on the verifier-frozen, auxiliary-head subfamily, where Medusa, Hydra, and EAGLE remain the closest reference points for our text-side drafter design.

Method	Drafter design	Sp. (reported)
ViSpec	vision-aware attention	$1.50\times$
MASSV	multimodal drafter	$1.46\times$
DREAM	cross-modal fusion	$1.45\times$
SpecVLM	vision-aware head	$1.50\times$
V5e-0	text-side linear heads	$1.89\times$

Table 1: Motivation-level comparison with recent VLM-SD methods. Source citations for each method appear in the surrounding text.

This section motivates a simpler VLM-SD design. We ask whether explicit visual processing inside the drafter is required for strong greedy VLM speculative decoding. We use two observations. First, recent vision-aware VLM-SD methods report speedups in a similar range despite using different visual-processing mechanisms. Second, the verifier's text-side hidden state h_t already provides a strong signal for next-token speculation. Together, these observations motivate treating VLM speculative drafting as language-side continuation modeling over verifier hidden states.

2.1 Vision-Aware Drafters and Reported Speedups

Table 1 summarizes reported average speedups for recent VLM-SD methods. The comparison is motivation-level rather than controlled, since the numbers come from different papers with different verifiers, prompts, hardware, and generation lengths. A controlled head-to-head comparison with ViSpec is provided in Section 4.3.

The prior methods explicitly add visual processing to the drafter through image-token attention, multimodal drafter adaptation, cross-modal fusion,

Verifier VLM	top-1	top-3	top-5
Qwen2-VL-2B	**0.880**	0.955	0.973
Qwen2-VL-7B	**0.925**	0.956	0.960
LLaVA-1.5-7B	**0.810**	0.851	0.867
LLaVA-1.6-Mistral	**0.797**	0.842	0.860

Table 2: Held-out next-token prediction accuracy using only the verifier's text-side hidden state.

or a vision-aware draft head. V5e-0 instead uses two text-side linear heads whose parameter count is determined only by the verifier hidden dimension; the exact parameter formula is given in Section 3.2. The comparison in Table 1 does not imply that vision-aware drafting is unnecessary in all settings. Concurrent benchmarks such as MMSpec (Shen et al., 2026) suggest that vision-aware drafting can be useful under more vision-grounded or larger-batch conditions. Our question is narrower: in the greedy VLM-SD setting evaluated in this paper, can a lightweight drafter obtain strong acceleration using only the verifier's text-side representation?

2.2 Text-Side Hidden States Provide a Strong Drafting Signal

We hypothesize that the verifier's text-side hidden state h_t already contains the multimodal context needed for next-token speculation. The verifier has processed both image and text tokens before producing h_t, and the same state is used by the frozen language-model head to predict the next token.

To test this hypothesis, we train a single linear continuation head:

$$\mathrm{Cont}(h_t, E[r^\star]) = W_Q\big(h_t + \alpha E[r^\star]\big), \quad (1)$$

where $r^\star$ is the verifier's most recently emitted token, $E[\cdot]$ is the frozen token embedding table, and α is a fixed scale. The output is projected through the frozen $\mathrm{lm_head}$ and trained with cross-entropy against the verifier's greedy next-token target using LLaVA-Instruct-150K (Liu et al., 2024b) paired with COCO images (Lin et al., 2014). No image tokens, vision encoder, cross-modal projection, or visual cross-attention are used in this drafter computation.

Table 2 shows that a text-side linear head predicts the verifier's greedy next token with high accuracy across representative VLMs. This supports using h_t as the drafter input and motivates V5e-0: a minimal drafter that leaves visual processing inside the verifier while performing speculative continuation modeling on the language-side state. The claim is intentionally scoped. Our TextVQA comparison in Section 4.3 shows that explicit visual access can still be beneficial on OCR-oriented inputs.

3 Method

V5e-0 is a minimalist self-speculative decoding framework for VLMs. It keeps the verifier frozen and attaches two linear continuation heads to the verifier's text-side hidden state. The drafter uses no image-token input, no visual cross-attention, and no verifier update. This differs from vision-aware VLM-SD methods (Kang et al., 2025; Ganesan et al., 2025; Hu et al., 2025; Huang et al., 2025), which route visual information into the drafter, and is closer in spirit to LLM-side auxiliary-head drafters such as Medusa (Cai et al., 2024), Hydra (Ankner et al., 2024), and EAGLE (Li et al., 2024b).

3.1 Problem Formulation

Let $\mathcal{V}$ denote a frozen VLM verifier. Given an image I and a text prefix $x_{<t}$, the vision encoder produces visual tokens

$$V = \phi_v(I) \in \mathbb{R}^{n_v \times D}. \quad (2)$$

The language backbone processes V and the embedded text prefix, and returns the final text-side hidden state

$$h_t = \mathcal{V}_{\text{backbone}}\big(V, E[x_{<t}]\big)[\text{last}] \in \mathbb{R}^D, \quad (3)$$

where $E[\cdot]$ is the frozen token embedding table. The verifier's greedy next token is

$$r_t^\star = \arg\max_x \mathrm{lm_head}(h_t)[x], \quad (4)$$

with frozen $\mathrm{lm_head}$.

In greedy speculative verification (Stern et al., 2018; Leviathan et al., 2023; Chen et al., 2023), a drafter proposes a tree of candidate tokens and the verifier evaluates the tree in one forward pass under a sibling-disjoint tree-attention mask (Miao et al., 2024; Chen et al., 2024). The emitted tokens are the longest candidate prefix matching the verifier's greedy path. Under exact arithmetic, this matches autoregressive greedy decoding.

3.2 V5e-0 Architecture

Figure 2 summarizes the V5e-0 inference and training pipeline. V5e-0 has three components: Free-Root, Cont, and Cont2.

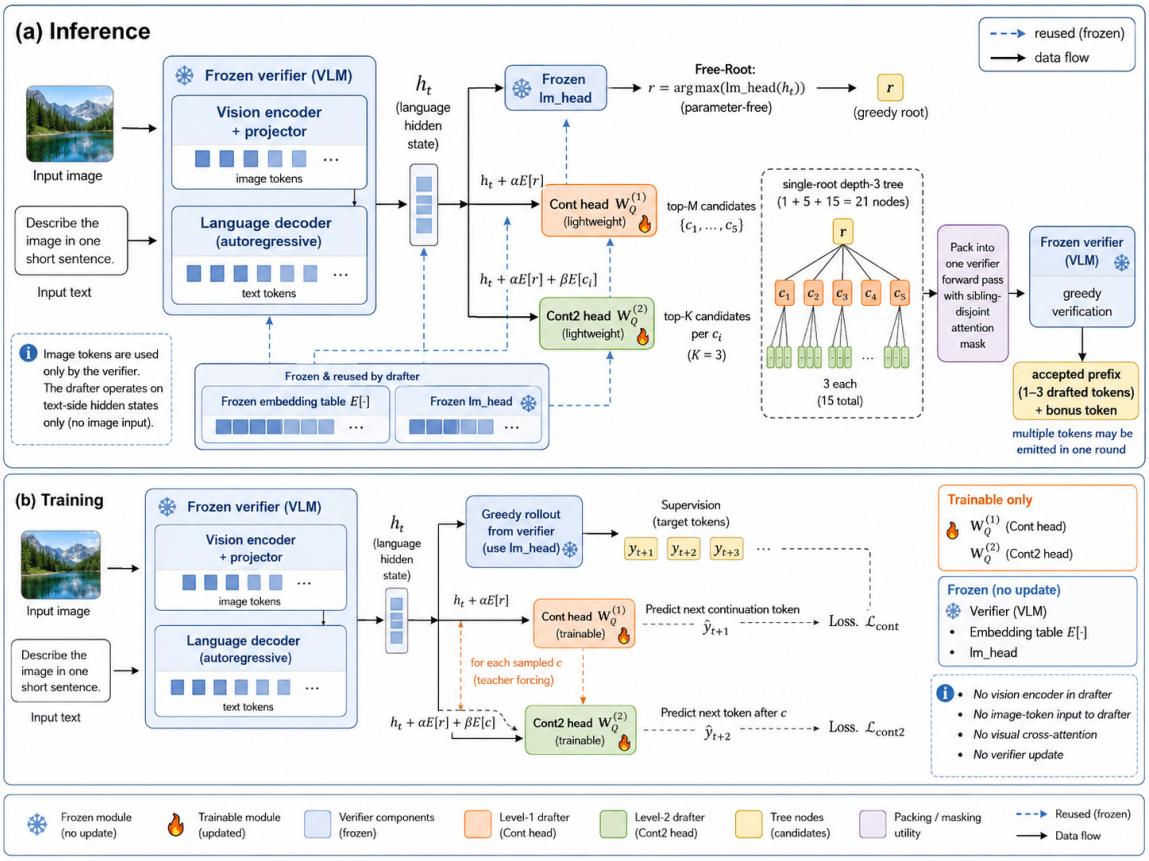

Figure 2: V5e-0 architecture for inference and training. **(a) Inference.** The frozen verifier produces h_t and its greedy argmax r. The Cont and Cont2 heads generate first- and second-level candidates using frozen embeddings and the frozen lm_head. The resulting single-root depth-3 tree is verified in one verifier forward pass, and the longest greedy-matched prefix is emitted. **(b) Training.** The continuation heads are trained from greedy verifier rollouts with cross-entropy losses; the verifier, embeddings, and lm_head remain frozen.

Free-Root. V5e-0 does not learn a root-prediction head. The root of each speculative round is the verifier's own greedy argmax:

$$r = \arg\max_x \text{lm_head}(h_t)[x]. \tag{5}$$

Thus, the root is accepted by construction under greedy verification and adds no root-prediction cost. This removes the need to model the root position and avoids allocating tree budget to root candidates that cannot be accepted under greedy verification.

Cont head. Given h_t and the Free-Root token r, the first continuation head predicts the next token:

$$z_1 = W_Q^{(1)}\big(h_t + \alpha E[r]\big) + b^{(1)}, \tag{6}$$

$$p_c(\cdot \mid h_t, r) = \text{softmax}\big(\text{lm_head}(z_1)\big). \tag{7}$$

The top-M tokens under p_c form the first-level candidates

$$\mathcal{C}_1 = \{c_1, \ldots, c_M\}. \tag{8}$$

By reading directly from h_t and the frozen embedding table, the drafter avoids maintaining a separate KV cache and reuses verifier-side state—a design principle also exploited, in a different form, by GliDe with a CaPE (Du et al., 2024).

Cont2 head. For each first-level candidate c_i, the second continuation head predicts one additional token:

$$z_2(c_i) = W_Q^{(2)}\big(h_t + \alpha E[r] + \beta E[c_i]\big) + b^{(2)}, \tag{9}$$

$$p_{c_2}(\cdot \mid h_t, r, c_i) = \text{softmax}\big(\text{lm_head}(z_2(c_i))\big). \tag{10}$$

The top-K tokens form

$$\mathcal{C}_2(c_i) = \{c_{i,1}, \ldots, c_{i,K}\}. \tag{11}$$

The dependence on $E[c_i]$ makes Cont2 sequentially conditioned on the selected first-level candidate, unlike parallel multi-head designs (Cai et al., 2024; Ankner et al., 2024).

Both heads project through the frozen lm_head, so the drafter stays in the verifier's vocabulary space without learning a separate vocabulary projection. We initialize $W_Q^{(1)}$ and $W_Q^{(2)}$ to the identity matrix, initialize both biases to zero, and use $\alpha = \beta = 30$.

Trainable parameters. The only trainable parameters are the two linear heads:

$$|\theta_{\text{V5e-0}}| = 2(D^2 + D). \tag{12}$$

For $D = 1536$, this gives 4.72M trainable parameters. The verifier, embedding table, and lm_head are reused without updates.

3.3 Single-Root Tree

V5e-0 forms a single-root depth-3 tree:

$$\begin{aligned}\mathcal{T} = \{r\} &\cup \{c_1, \ldots, c_M\} \\ &\cup \left\{ c_{i,j} \,\middle|\, \begin{smallmatrix} 1 \le i \le M \\ 1 \le j \le K \end{smallmatrix} \right\}.\end{aligned} \tag{13}$$

The tree contains $1 + M + MK$ candidate nodes. We use $M = 5$ and $K = 3$ by default, yielding 21 candidates per round.

The single-root design follows from Free-Root. Since r is the verifier's greedy root, additional root candidates cannot be accepted under greedy verification. This contrasts both with fixed multi-root templates derived from blockwise parallel decoding (Stern et al., 2018) and with adaptive draft-tree expansion such as OPT-Tree (Wang et al., 2024a) and EAGLE-2/3 (Li et al., 2024a, 2025), which grow or reshape the draft tree per round to maximize expected acceptance.

Proposition 1 (Single-Root Budgeting)**.** *Let $\{r_1, \ldots, r_N\}$ be sibling-disjoint root candidates and suppose $r_1 = r_t^\star$ is the verifier's greedy token. Under greedy verification,*

$$\Pr\big[\text{accept}\big(\text{desc}(r_i)\big)\big] = 0, \qquad \forall i \ge 2, \tag{14}$$

where $\text{desc}(r_i)$ denotes descendants of root candidate r_i.

Proof. Greedy verification accepts only the root candidate matching $r_t^\star$. Since $r_1 = r_t^\star$, every other root candidate r_i with $i \ge 2$ is rejected at the root position. With sibling-disjoint tree attention, descendants of a rejected root cannot lie on an accepted path, so they contribute zero accepted tokens. □

This result only applies to greedy verification and sibling-disjoint tree attention. Sampling-based decoding uses a different acceptance rule.

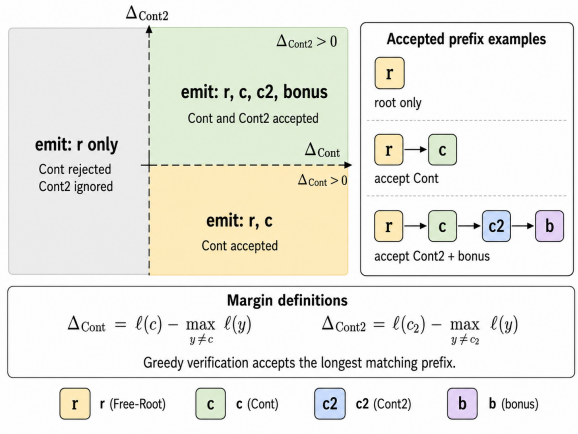

Figure 3: Acceptance regions for one V5e-0 speculative round. Positive verifier margins for Cont and Cont2 determine whether V5e-0 emits r, r, c, or r, c, c_2 plus a verifier bonus token b.

3.4 Greedy Verification

After constructing $\mathcal{T}$, V5e-0 packs all nodes into one verifier forward pass. The verifier logits define a greedy path through the tree, and V5e-0 emits only the longest drafted prefix matching that path.

For a candidate chain $r \to c \to c_2$, define the verifier margins

$$\Delta_{\text{Cont}} = \ell_r(c) - \max_{y \neq c} \ell_r(y), \tag{15}$$

$$\Delta_{\text{Cont2}} = \ell_c(c_2) - \max_{y \neq c_2} \ell_c(y), \tag{16}$$

where $\ell_r(\cdot)$ and $\ell_c(\cdot)$ are verifier logits at the root and Cont positions. If $\Delta_{\text{Cont}} < 0$, V5e-0 emits only r. If only Cont is accepted, it emits r, c. If both Cont and Cont2 are accepted, it emits r, c, c_2 plus a verifier bonus token b from the deepest accepted node.

3.5 Training

V5e-0 is trained from greedy rollouts of the frozen verifier. It uses no external annotations, auxiliary teacher, feature distillation, or verifier fine-tuning.

Data. Training prompts are drawn from LLaVA-Instruct-150K (Liu et al., 2024b) paired with COCO images (Lin et al., 2014). For each verifier, we sample 200 prompts and roll out 32 greedy steps per prompt, yielding 6,400 training records. At position t, we record

$$\mathcal{R}_t = (h_t, r_t, c_t, c_{2,t}), \tag{17}$$

where

$$r_t = \arg\max_x \mathrm{lm_head}(h_t)[x], \tag{18}$$

$$c_t = \arg\max_x \mathrm{lm_head}(h_{t+1})[x], \tag{19}$$

$$c_{2,t} = \arg\max_x \mathrm{lm_head}(h_{t+2})[x]. \tag{20}$$

Here h_{t+1} and h_{t+2} are hidden states after appending r_t and r_t, c_t, respectively.

Objective. The two heads are trained with token-level cross-entropy:

$$\mathcal{L}_{\mathrm{Cont}} = -\frac{1}{|\mathcal{R}|} \sum_t \log p_c(c_t \mid h_t, r_t), \tag{21}$$

$$\mathcal{L}_{\mathrm{Cont2}} = -\frac{1}{|\mathcal{R}|} \sum_t \log p_{c_2}(c_{2,t} \mid h_t, r_t, c_t). \tag{22}$$

We train Cont first and then train Cont2 conditioned on the Cont target, matching the inference-time dependency in Equation 9.

Optimization. Both heads are optimized with AdamW using learning rate 5×10^{-4}, weight decay 0.01, and batch size 8. Each head is trained for 100 epochs in bf16 mixed precision. We train $W_Q^{(1)}$ first, freeze it, and then train $W_Q^{(2)}$. Training takes approximately one minute per verifier on a single A100 80GB GPU.

3.6 Inference

At inference time, V5e-0 repeats one speculative round until generation terminates. The verifier provides the current hidden state h_t and greedy logits. V5e-0 takes the verifier argmax as the Free-Root token, generates M Cont candidates and K Cont2 candidates per Cont candidate, and packs the resulting single-root tree into one verifier forward pass.

The verifier then selects the greedy path through the packed tree. V5e-0 emits the longest candidate prefix matching that path, optionally followed by the verifier bonus token at the deepest accepted node. If the first continuation is rejected, the round emits only the Free-Root token and proceeds to the next round.

4 Experiments

We evaluate V5e-0 on wall-clock speedup across fifteen VLMs, controlled baselines, vision-grounded task utility, design ablations, and output fidelity.

4.1 Setup

Training and held-out evaluation prompts are drawn from LLaVA-Instruct-150K (Liu et al., 2024b) with COCO images (Lin et al., 2014): 200 training prompts $\times$32 greedy verifier rollouts (6,400 records) per verifier, 64-token greedy evaluation on a disjoint split (n=100 prompts; n=30 for the four recently added models marked * in Table 3). We report wall-clock speedup $\mathrm{sp} = T_{\mathrm{AR}}/T_{\mathrm{V5e\text{-}0}}$ and tokens per V5e-0 round (TPI), following recent speculative-decoding benchmarks (Xia et al., 2024). All measurements use a single A100 80GB with bf16 mixed precision and each verifier's default attention backend, so V5e-0's speedup is isolated from orthogonal kernel-level accelerations such as FlashAttention (Dao et al., 2022). Prompt IDs, seeds, and reproduction scripts are released with the code.

4.2 Main Wall-clock Results

Table 3 shows a mean speedup of $\mathbf{1.89}\times$ over AR decoding (per-model range 1.37–2.52$\times$). Speedup tracks TPI: V5e-0 accelerates decoding by emitting multiple accepted tokens per verifier pass without modifying the verifier, complementary to alternative parallel-decoding paradigms such as PEARL (Liu et al., 2024c) and serving-system schemes such as SpecExec (Svirschevski et al., 2024).

4.3 Controlled Baseline Comparisons

We compare V5e-0 with a vision-aware VLM-SD baseline and an EAGLE-style lightweight drafter under matched prompts, hardware, generation length, and AR baseline.

ViSpec. Table 4 reports the comparison on LLaVA-1.5-7B + TextVQA (Singh et al., 2019).

Method	Throughput	Speedup
AR baseline	36.5	1.000$\times$
ViSpec (Kang et al., 2025)	59.4	1.670$\times$
V5e-0	49.7	$1.362 \pm 0.170\times$

Table 4: Controlled comparison on LLaVA-1.5-7B + TextVQA.

ViSpec is faster on TextVQA, indicating that explicit visual access can help on OCR-oriented inputs and supporting a domain-dependent comparison between text-side and vision-aware drafting.

Model	Throughput (t/s)		Acceleration	
	AR	V5e-0	Sp (mean±std)	TPI
Qwen2-VL-2B	35.37	67.56	1.911 ± 0.312	2.34
Qwen2-VL-7B	35.84	78.83	2.200 ± 0.331	2.70
Qwen2.5-VL-7B	35.82	56.44	1.575 ± 0.339	1.92
Qwen3-VL-4B*	26.43	41.99	1.561 ± 0.437	1.93
LLaVA-1.5-7B	36.41	68.37	1.878 ± 0.324	2.16
LLaVA-1.6-Mistral-7B	35.67	60.42	1.694 ± 0.360	2.06
LLaVA-1.6-Vicuna-7B	35.18	48.24	1.372 ± 0.281	1.81
LLaVA-OneVision-7B	36.96	78.81	2.133 ± 0.361	2.67
Idefics2-8B	35.44	56.20	1.586 ± 0.088	2.05
PaliGemma-3B	54.99	129.57	2.356 ± 0.172	**2.92**
PaliGemma2-3B	30.75	71.11	2.312 ± 0.395	2.61
PaliGemma2-10B	19.26	48.58	$\mathbf{2.519 \pm 0.241}$	2.83
GLM-4V-9B*	26.98	55.79	2.068 ± 0.368	2.49
InternVL3.5-8B*	21.60	36.05	1.668 ± 0.241	1.88
Pixtral-12B*	27.54	41.09	1.492 ± 0.229	1.86

Table 3: Wall-clock performance of V5e-0 across fifteen VLMs. Throughput is measured in tokens per second; Sp is per-prompt speedup over AR; TPI is the mean number of tokens emitted per V5e-0 round. Models marked * are evaluated with n=30 prompts; the remaining models use n=100. **Bold** denotes the best value in the Speedup and TPI columns.

EAGLE-style drafter. Table 5 compares V5e-0 with EAGLE-VLM Light (Li et al., 2024b); the latter is a recent line that also includes representation-harmonization variants such as HASS (Zhang et al., 2024b).

Method	Params	Cont top-1	Speedup
EAGLE-VLM Light	28.3M	0.651	1.211×
V5e-0	**4.72M**	**0.880**	1.362×

Table 5: Controlled comparison with an EAGLE-style VLM-adapted lightweight drafter on Qwen2-VL-2B.

V5e-0 obtains higher continuation accuracy and speedup with fewer trainable parameters, supporting direct token-space continuation through the frozen lm_head.

4.4 Vision-Grounded Task Utility

Table 6 reports answer accuracy, speedup, and exact text match on TextVQA (Singh et al., 2019), OCRBench (Liu et al., 2024d), DocVQA (Mathew et al., 2021), and ChartQA (Masry et al., 2022).

Benchmark	n	Speedup	AR Acc.	V5e-0 Acc.	Match
TextVQA	100	1.41×	86.0%	86.0%	96.0%
OCRBench	50	1.461 ± 0.211×	96.0%	96.0%	30.0%
DocVQA	50	1.250 ± 0.503×	28.0%	28.0%	28.0%
ChartQA	50	1.158 ± 0.336×	38.0%	40.0%	14.0%

Table 6: Task-level utility on vision-grounded benchmarks with Qwen2-VL-2B. Match denotes exact text match between V5e-0 and AR outputs.

V5e-0 preserves answer-level accuracy on all evaluated benchmarks even when exact text match is low; speedup decreases on OCR/document/chart inputs, so distribution shift mainly reduces acceleration rather than task accuracy.

4.5 Ablation Studies

We ablate four design choices: drafter-side visual input, Free-Root, sequential Cont2 conditioning, and tree configuration.

4.5.1 Vision input to the drafter

Variant	Vision input	TPI	Sp (mean±std)
Qwen2-VL-2B			
V5e-0	–	2.12	1.751 ± 0.388
V5e-2	mean-pool	2.13	1.765 ± 0.373
V5d	cross-attn	1.62	1.350 ± 0.325
LLaVA-1.5-7B			
V5e-0	–	2.08	1.807 ± 0.316
V5e-2	mean-pool	2.11	1.832 ± 0.296
V5d	cross-attn	1.37	1.128 ± 0.115

Table 7: Vision-input ablation with end-to-end wall-clock speedup.

Mean-pooled visual features are comparable to the text-side drafter; cross-attention reduces TPI and speedup under the same lightweight-head training budget.

4.5.2 Root prediction

Root-prediction strategy	top-1
Free-Root	**1.000**
Untrained identity head	1.000
Linear head, CE-trained	0.818

Table 8: Root-prediction ablation on Qwen2-VL-2B.

Free-Root is correct by construction (it reuses the verifier's greedy argmax); a learned root head is less reliable because it approximates a decision the verifier already computes.

4.5.3 Sequential vs. parallel Cont2

Verifier	Variant	Cont2 top-1	Speedup
Qwen2-VL-2B	Parallel	0.610	$1.478\times$
	Sequential	**0.880**	$1.499\times$
Qwen2-VL-7B	Parallel	0.869	$2.159\times$
	Sequential	**0.925**	$2.222\times$
LLaVA-1.5-7B	Parallel	0.602	$1.488\times$
	Sequential	**0.810**	$1.467\times$

Table 9: Sequential vs. parallel Cont2 ablation across three VLMs.

Sequential conditioning improves Cont2 top-1 accuracy on all three VLMs; the wall-clock effect is verifier-dependent because Cont2 helps only when the preceding Cont token is also accepted.

4.5.4 Tree configuration

Configuration	Size	TPI	Speedup
Depth-3 $(M{=}5, K{=}3)$	**21**	2.49	$2.04\times$
Depth-4 $(M{=}5, K{=}2, L{=}1)$	26	**2.62**	$2.09\times$
Depth-4 $(M{=}5, K{=}3, L{=}1)$	36	2.61	$2.00\times$
Depth-4 $(M{=}3, K{=}2, L{=}2)$	22	2.49	$2.03\times$

Table 10: Depth-4 tree ablation on Qwen2-VL-2B.

Depth-4 slightly increases TPI but adds at most $+0.05\times$ speedup while enlarging the tree; we therefore use the depth-3 tree as the default speed–complexity trade-off.

4.6 Output Fidelity under Greedy Verification

Greedy verification preserves the verifier's greedy path under exact arithmetic; under bf16 the tree-forward and token-by-token AR paths can diverge when top logits are nearly tied. Across 450 generations V5e-0 obtains $\approx 90\%$ token-level match and 0.94 ROUGE-L against AR, and a precision sweep shows that fp16 matches the fp32 greedy path on the tested Qwen2-VL-2B subset while retaining most of the default throughput. We therefore report empirical output similarity rather than byte-identical equivalence under bf16.

Limitations

V5e-0 is limited to greedy speculative verification. Its Free-Root design and single-root tree allocation rely on the verifier's greedy acceptance rule, so sampling-based decoding requires a different acceptance analysis. The drafter is also trained per verifier and is not designed for cross-architecture transfer. In addition, V5e-0 does not guarantee byte-identical outputs under bf16 execution because tree-forward and token-by-token AR execution can diverge numerically when verifier logits are nearly tied.

Our evaluation also shows a domain-dependent trade-off. V5e-0 preserves task accuracy on the evaluated OCRBench subset, but its speedup is lower than on LLaVA-Instruct-style prompts, and ViSpec is faster on TextVQA. Thus, V5e-0 should be viewed as a strong minimalist baseline for greedy VLM speculative decoding, not as a universal replacement for vision-aware drafters. Broader MMSpec-style evaluation, larger-scale batched serving, multilingual prompts, long-form generation, multi-image inputs, and video settings remain future work.

5 Conclusion

We presented V5e-0, a minimalist self-speculative decoding framework for VLMs that drafts from the verifier's text-side hidden state without image-token input, visual cross-attention, or verifier updates. Across fifteen VLMs, V5e-0 achieves a mean wall-clock speedup of $1.89\times$ over autoregressive decoding in the evaluated greedy setting.

Additional experiments clarify the scope of this result. V5e-0 preserves AR task accuracy on OCRBench with a lower but meaningful speedup, while ViSpec is faster on TextVQA, showing that vision-aware drafters can remain advantageous on OCR-heavy inputs. Overall, V5e-0 establishes a simple and practical baseline for greedy VLM speculative decoding, and provides a reference point for future hybrid methods that combine text-side efficiency with vision-aware drafting.

References

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Yusuke Magaki, and Jonathan Frankle. 2024. Hydra: Sequentially-dependent draft heads for medusa decoding. *arXiv preprint arXiv:2402.05109*.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st International Conference on Machine Learning*.

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

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yu-Hsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. 2024. Sequoia: Scalable and robust speculative decoding. In *Advances in Neural Information Processing Systems*.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359.

Cunxiao Du, Jing Jiang, Xu Yuanchen, Jiawei Wu, Sicheng Yu, Yongqi Li, Shenggui Li, Kai Xu, Liqiang Nie, Zhaopeng Tu, and Yang You. 2024. GliDe with a CaPE: A low-hassle method to accelerate speculative decoding. In *Proceedings of the 41st International Conference on Machine Learning*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Mugilan Ganesan, Shane Segal, Ankur Aggarwal, Nish Sinnadurai, Sean Lie, and Vithursan Thangarasa. 2025. MASSV: Multimodal adaptation and self-data distillation for speculative decoding of vision-language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 12265–12276, Suzhou, China. Association for Computational Linguistics.

Yunhai Hu, Tianhua Xia, Zining Liu, Rahul Raman, Xingyu Liu, Bo Bao, Eric Sather, Vithursan Thangarasa, and Sai Qian Zhang. 2025. DREAM: Drafting with refined target features and entropy-adaptive cross-attention fusion for multimodal speculative decoding. In *Advances in Neural Information Processing Systems*.

H. Huang and 1 others. 2025. Fast speculative decoding in vision-language models. *arXiv preprint arXiv:2509.11815*.

Jialiang Kang, Han Shu, Wenshuo Li, Yingjie Zhai, and Xinghao Chen. 2025. ViSpec: Accelerating vision-language models with vision-aware speculative decoding. *arXiv preprint arXiv:2509.15235*.

Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. 2024. CLLMs: Consistency large language models. In *Proceedings of the 41st International Conference on Machine Learning*. PMLR.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, pages 19274–19286. PMLR.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19730–19742. PMLR.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024a. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7421–7432, Miami, Florida, USA. Association for Computational Linguistics.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer.

Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Kai Han, and Yunhe Wang. 2024a. Kangaroo: Lossless self-speculative decoding via double early exiting. *arXiv preprint arXiv:2404.18911*.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024b. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26296–26306.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pages 34892–34916.

Tianyu Liu, Yun Li, Qitan Lv, Kai Liu, Jianchen Zhu, Winston Hu, and Xiaohui Sun. 2024c. PEARL: Parallel speculative decoding with adaptive draft length. *arXiv preprint arXiv:2408.11850*.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. 2024d. OCRBench: On the hidden mystery of OCR in large multimodal models. *Science China Information Sciences*, 67(12):220102.

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

Minesh Mathew, Dimosthenis Karatzas, and C.V. Jawahar. 2021. DocVQA: A dataset for VQA on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2200–2209.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2024. SpecInfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (ASPLOS '24)*, pages 932–949, New York, NY, USA. Association for Computing Machinery.

Hui Shen, Xin Wang, Ping Zhang, Yunta Hsieh, Qi Han, Zhongwei Wan, Ziheng Zhang, Jingxuan Zhang, Jing Xiong, Ziyuan Liu, and 1 others. 2026. MMSpec: Benchmarking speculative decoding for vision-language models. *arXiv preprint arXiv:2603.14989*.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards VQA models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8317–8326.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, volume 31.

Ruslan Svirschevski, Avner May, Zhuoming Chen, Beidi Chen, Zhihao Jia, and Max Ryabinin. 2024. SpecExec: Massively parallel speculative decoding for interactive LLM inference on consumer devices. *arXiv preprint arXiv:2406.02532*.

Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. 2024a. OPT-Tree: Speculative decoding with adaptive draft tree structure. *arXiv preprint arXiv:2406.17276*.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, and 1 others. 2024b. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7655–7671, Bangkok, Thailand. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Gao, Chengyuan Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2024a. Draft & verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11263–11282, Bangkok, Thailand. Association for Computational Linguistics.

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. 2024b. Learning harmonized representations for speculative sampling. *arXiv preprint arXiv:2408.15766*.

A Additional Experimental Details

Round	Prefix state	Packed candidate tree input	Verifier decision / emitted tokens
1	`[image, prompt]`	Root: `The`. Cont candidates: {`image`, `picture`, `scene`, `photo`, `view`}. For `image`, Cont2 candidates: {`fea-`, `shows`, `contains`}.	Verifier path matches `The` → `image` → `fea-` plus bonus `tures`. **Emit:** `The image fea- tures`.
2	`The image features`	Root: `a`. Cont candidates: {`gi-`, `person`, `large`, `young`, `tall`}. For `gi-`, Cont2 candidates: {`raffe`, `ant`, `rl`}.	Verifier path matches `a` → `gi-` → `raffe` plus bonus `standing`. **Emit:** `a gi- raffe standing`.
3	`The image features a giraffe standing`	Root: `in`. Cont candidates: {`near`, `next`, `on`, `by`, `at`}. For `near`, Cont2 candidates: {`a`, `the`, `some`}.	Verifier matches root only; `near` is not the greedy continuation. **Emit:** `in`.

Table A1: Illustrative V5e-0 inference trace over three rounds.

Table A1 illustrates one V5e-0 decoding trace. Each round packs a single-root candidate tree into one verifier forward pass and emits only the longest prefix matching the verifier's greedy path. The remaining appendix provides implementation notes, tree-design analyses, baseline comparisons, robustness checks, scaling analyses, and numerical-fidelity measurements.

A.1 Setup and Reproducibility Notes

All experiments follow the protocol in Section 4.1. Model-specific wrappers for recently released VLMs only adapt verifier APIs and do not change the V5e-0 drafter, tree construction, or greedy verification rule. We release training and inference code, prompt IDs, random seeds, per-verifier configuration files, and scripts for reproducing the fifteen-VLM sweep. The training split uses seed 43, and the held-out evaluation split uses seed 999.

A.2 Tree-Attention Implementation

V5e-0 verifies a candidate tree by packing candidate nodes into a single verifier forward pass with a sibling-disjoint attention mask. For Qwen-VL-family models, we preserve the model's multimodal rotary position encoding when constructing packed tree inputs. For the remaining VLMs, we use the model's standard RoPE or positional encoding path. We use `sdpa` attention when the verifier implementation supports the required mask and fall back to eager attention for GLM-4V and InternVL3.5. All experiments use HuggingFace-style KV-cache objects, including `DynamicCache` where required by the verifier implementation.

Verifier	Static $(16, 3)$	Single-root $M{=}15$	V5e-0 $(5, 3)$
Qwen2-VL-2B	1.375	1.560	1.911
Qwen2-VL-7B	1.493	1.590	2.200
LLaVA-1.5-7B	1.377	1.570	1.878
LLaVA-1.6-Mistral	1.287	1.420	1.694
Mean	1.383	1.535	**1.921**
Δ vs. static	–	$+10.4\%$	$+38.9\%$

Table A2: Tree-shape progression on four VLMs.

A.3 Parameter Accounting for ViSpec

We omit ViSpec parameter counts from the controlled throughput table because parameter accounting differs by convention. The source paper reports the additional transformer block size, our matched protocol counts the active drafter module used during inference, and the released checkpoint contains additional weights that are not all active in the measured path. Accordingly, we use the ViSpec comparison only as a controlled throughput comparison under matched prompts, hardware, generation length, and AR baseline.

A.4 Tree Design Analyses

We compare V5e-0 with fixed tree baselines. Static $(16, 3)$ denotes a multi-root tree with 16 root candidates and 3 continuation candidates under each root, yielding 64 candidate nodes in total. Single-root $M{=}15$ denotes a depth-2 tree with the Free-Root token and 15 first-level continuation candidates. V5e-0 uses the default single-root depth-3 tree with $(M, K) = (5, 3)$ and 21 candidate nodes.

The default single-root depth-3 tree provides the best speed–complexity trade-off. Increasing tree size beyond the default does not reliably improve wall-clock speedup because verifier-side tree-

M	K	Size	TPI	Speedup
3	2	10	2.21	1.80×
3	3	13	2.32	1.89×
5	1	11	2.18	1.85×
5	2	16	2.39	1.97×
5	**3**	**21**	**2.49**	2.04×
5	4	26	2.51	2.02×
7	3	29	2.51	2.00×
10	1	21	2.31	2.01×
10	3	41	2.55	1.97×

Table A3: Full (M, K) sweep on Qwen2-VL-2B. The default configuration is highlighted.

Configuration	Size	TPI	d_2 rate	d_3 rate
Depth-3 (M=5, K=3)	21	2.49	–	–
Depth-4 (M=5, K=2, L=1)	26	2.62	0.74	0.53
Depth-4 (M=5, K=3, L=1)	36	2.61	0.72	0.53
Depth-4 (M=3, K=2, L=2)	22	2.49	0.67	0.46

Table A4: Full depth-4 tree ablation on Qwen2-VL-2B.

Setting	Tree mix	Speedup	Δ vs. static
Conf. $\tau = 0.5$	92.5% small	1.406 ± 0.129	$-0.51\times$
Conf. $\tau = 0.7$	82.2% small	1.467 ± 0.132	$-0.44\times$
Conf. $\tau = 0.9$	69.3% small	1.537 ± 0.132	$-0.38\times$
Entropy $(0.3, 1.2)$	27/27/46	1.845 ± 0.293	$-0.07\times$
Entropy $(0.5, 1.5)$	33/27/41	1.843 ± 0.292	$-0.07\times$
Entropy $(0.8, 2.0)$	42/30/27	1.874 ± 0.282	$-0.04\times$
Static depth-3	full	1.911×	–

Table A5: Dynamic-tree ablations on Qwen2-VL-2B. Entropy rows report small/medium/full tree percentages.

forward cost grows faster than the accepted-token gain.

A.5 Dynamic Tree Variants

We evaluate two lightweight dynamic-tree variants. The first uses drafter confidence to switch to a smaller tree. The second uses verifier root entropy to choose among small, medium, and full trees. These experiments are sanity checks rather than full dynamic speculative decoding methods.

Confidence-only gating underperforms because it switches to small trees too often and loses Cont2 and bonus-token opportunities. Entropy-aware gating closes most of the gap to the static depth-3 tree, but it still does not outperform the default.

A.6 Sequential Cont2 Conditioning

Table A6 compares the proposed sequential Cont2 head with a parallel variant. Sequential Cont2 conditions on the selected Cont token through $E[c]$, while the parallel variant predicts the second continuation from (h_t, r) alone.

Verifier	Variant	Cont2 top-1	Speedup
Qwen2-VL-2B	Parallel	0.610	1.478×
	Sequential	**0.880**	1.499×
Qwen2-VL-7B	Parallel	0.869	2.159×
	Sequential	**0.925**	2.222×
LLaVA-1.5-7B	Parallel	0.602	1.488×
	Sequential	**0.810**	1.467×

Table A6: Sequential vs. parallel Cont2 ablation across three VLMs.

Method	Params	Cont top-1	Speedup
EAGLE-VLM Light	28.3M	0.651	1.211×
V5e-0	**4.72M**	**0.880**	1.362×

Table A7: Controlled comparison with an EAGLE-style VLM-adapted lightweight drafter on Qwen2-VL-2B.

Model	Drafter	lm_head	Verify	Other	Total
Qwen2-VL-2B	0.12	2.62	29.76	2.73	35.22
Qwen2-VL-7B	0.12	1.53	29.70	2.67	34.02
LLaVA-1.5-7B	0.14	0.51	27.20	3.09	30.94

Table A8: Per-round runtime breakdown in milliseconds.

Model	AR	V5e-0	Overhead
Qwen2-VL-2B	5.53	5.58	+1.0%
Qwen2-VL-7B	19.10	19.20	+0.5%
LLaVA-1.5-7B	15.85	16.01	+1.0%

Table A9: Peak GPU memory in GB.

Sequential conditioning improves Cont2 top-1 accuracy on all three VLMs. Its wall-clock effect is verifier-dependent because Cont2 contributes only when the preceding Cont candidate is also accepted.

A.7 EAGLE-VLM Light Comparison

Table A7 compares V5e-0 with an EAGLE-style lightweight VLM-adapted drafter on Qwen2-VL-2B.

In this controlled comparison, V5e-0 is smaller, more accurate at continuation prediction, and faster.

A.8 System Overheads

Tables A8 and A9 report runtime and memory overheads on representative VLMs.

The per-round latency is dominated by verifier tree-forward computation. Peak memory overhead remains at or below 1.0% in the tested models.

Batch	AR t/s	V5e-0 t/s	Speedup	AR scaling
1	37.16 ± 0.05	62.68 ± 10.65	1.687 ± 0.286	$1.00\times$
2	73.69 ± 1.58	129.56 ± 17.47	1.757 ± 0.226	$1.98\times$
4	147.59 ± 1.08	267.91 ± 52.58	1.815 ± 0.355	$3.97\times$
8	294.11 ± 1.37	506.11 ± 78.46	1.722 ± 0.272	$7.92\times$
16	582.68 ± 5.63	986.82 ± 130.05	1.695 ± 0.232	$15.69\times$

Table A10: Batch-size scaling on Qwen2-VL-2B with replicated batches, n=5 prompts, and 64-token greedy generation.

Length	V5e-0 t/s	Speedup	TPI
32	78.10 ± 12.85	2.119 ± 0.348	2.62
64	70.88 ± 11.51	1.924 ± 0.313	2.36
128	64.98 ± 9.98	1.765 ± 0.272	2.17
256	63.52 ± 10.55	1.726 ± 0.288	2.12

Table A11: Generation-length scaling on Qwen2-VL-2B with n=15 prompts.

Top-p	Speedup
1.0	$1.170\times$
0.9	$1.182\times$
0.7	$1.228\times$

Table A12: Preliminary top-p sampling results.

A.9 Batch and Length Scaling

Table A10 reports batch-size scaling on Qwen2-VL-2B using replicated batches. This experiment is a throughput-scaling sanity check rather than a full heterogeneous serving benchmark.

V5e-0 maintains a 1.69–$1.82\times$ speedup across the tested batch sizes.

Speedup decreases as generation length increases, tracking the decrease in TPI. V5e-0 nevertheless retains a $1.73\times$ speedup at 256 generated tokens.

A.10 Top-p Sampling Preliminary

Table A12 gives a preliminary top-p sampling check. This experiment is not used for the main claim, which is restricted to greedy verification.

The speedup remains modest under top-p sampling. This supports keeping the theoretical analysis focused on greedy verification.

A.11 Training-Size Sensitivity

Table A13 varies the number of training prompts while keeping 32 greedy rollout steps per prompt. The default setting uses 200 prompts and 6,400 records.

Speedup saturates near the default 6,400 training records. Doubling the training set to 12,800 records changes speedup by only $0.003\times$.

Prompts	Records	Cont top-1	Speedup	TPI
50	1,600	0.637	1.761 ± 0.331	2.14
100	3,200	0.592	1.852 ± 0.363	2.23
200	6,400	0.608	1.925 ± 0.312	2.36
400	12,800	0.635	1.922 ± 0.273	2.38

Table A13: Training-size sensitivity on Qwen2-VL-2B.

Model	D	Params	Drafter ms	Share
Qwen2-VL-2B	1536	4.7M	0.80	2.3%
PaliGemma-3B	2048	8.4M	1.42	6.3%
Qwen3-VL-4B	2560	13.1M	2.22	4.8%
Qwen2-VL-7B	3584	25.7M	4.36	12.7%
LLaVA-1.5-7B	4096	33.6M	5.69	18.0%
PaliGemma2-10B	5120	52.4M	8.89	15.3%
Pixtral-12B	5120	52.4M	8.89	19.6%

Table A14: Hidden-dimension scaling analysis. Drafter latency is measured on Qwen2-VL-2B and extrapolated as D^2 for larger hidden sizes.

Verifier	Rounds	Root only	3 tokens	4 tokens
Qwen2-VL-2B	557	26.6%	18.1%	**55.3%**
LLaVA-1.5-7B	598	33.1%	20.9%	**46.0%**

Table A15: Accepted-prefix distribution over V5e-0 rounds. "4 tokens" denotes root, Cont, Cont2, and bonus emission.

A.12 Hidden-Dimension Scaling

V5e-0 uses two $D \times D$ linear heads, so drafter cost increases with the verifier hidden dimension. Table A14 estimates the drafter latency share by scaling the measured Qwen2-VL-2B drafter latency as D^2.

The estimated drafter share remains below 20% up to $D = 5120$, suggesting that the two linear heads are not the dominant runtime component even for larger hidden dimensions.

A.13 Accepted-Prefix Distribution

Table A15 reports round-level emission lengths for two representative VLMs.

More than half of Qwen2-VL-2B rounds emit the full depth-3 path plus bonus. LLaVA-1.5-7B has a lower full-emission rate, consistent with its lower TPI and speedup.

A.14 Embedding-Scale Sensitivity

Table A16 sweeps the shared embedding scale on Qwen2-VL-2B and LLaVA-1.5-7B. The goal is to check whether the continuation head is sensitive to the exact scale used when adding token embeddings to h_t.

Scale	Qwen2-VL-2B		LLaVA-1.5-7B	
$\alpha = \beta$	top-1	top-3	top-1	top-3
10	0.595	0.754	0.452	0.572
30	0.629	0.768	0.484	0.594
100	**0.639**	0.764	**0.514**	0.608
300	0.638	0.759	0.505	**0.609**

Table A16: Embedding-scale sensitivity for the Cont head on held-out records.

The default scale $\alpha = \beta = 30$ is not individually tuned per verifier. Accuracy changes moderately across the tested range, and no setting changes the overall conclusion.

A.15 Text-Only Transfer

Table A17 applies the same V5e-0 drafter to two text-only LLMs. Two VLM rows are included only as reference points.

Model	α_c	$\alpha_{c_2\|c}$	TPI	Speedup
Text-only LLMs				
Qwen3-4B (Yang et al., 2025)	0.41	0.32	1.54	1.39×
Llama-3.2-3B (Dubey et al., 2024)	0.45	0.34	1.60	1.44×
VLM reference rows				
Qwen2-VL-2B	0.80	0.69	2.34	1.91×
LLaVA-1.5-7B	0.78	0.59	2.16	1.88×

Table A17: Cross-modality transfer to text-only LLMs.

The text-only models obtain lower acceptance rates and TPI than the VLM references, but still retain non-trivial speedup. This supports the view that V5e-0 acts as a language-side continuation drafter rather than a vision-specific module.

A.16 Precision and Numerical Fidelity

Table A18 summarizes precision sensitivity and tree-verification fidelity on Qwen2-VL-2B. The precision rows measure agreement with the fp32 greedy path, while the tree rows report token-level match under different speculative tree configurations.

Check	Setting	Note	Match
Precision sensitivity			
Precision	bf16	default	75.1%
Precision	fp16	97% throughput	100.0%
Precision	fp32	reference	100.0%
Tree-verification fidelity			
Tree	M=3, d=2	size 4	70.47%
Tree	M=15, d=2	size 16	82.81%
Tree	M=5, K=3, d=3	size 21	88.44%
Tree	Static $(16, 3)$, d=2	size 64	88.91%
Tree	fp32 check	size 21	100.0%

Table A18: Precision sensitivity and numerical fidelity on Qwen2-VL-2B. The fp32 tree-verification check is measured on 10 held-out prompts.

The fp16 run matches the fp32 greedy path on the tested subset while retaining most of the default throughput. The lower bf16 agreement indicates that part of the observed mismatch is tied to mixed-precision numerical sensitivity. Across the full fifteen-VLM bf16 evaluation, V5e-0 obtains approximately 90% token-level match and 0.94 ROUGE-L against AR outputs.