VTD: VISIBLE-TEXT DRAFTING FOR SPECULATIVE DECODING IN VISION-LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generating HTML or Markdown from images often requires vision-language models to produce long sequences, making autoregressive decoding a major source of latency. Retrieval drafters can accelerate generation without training, but their text sources may omit content visible in the image. At steps where dynamic retrieval accepts no draft descendant, only 2.3% of the target's next three-token spans have a matching occurrence beginning at an earlier position in the same output. This limited recurrence motivates an additional retrieval source. We introduce VTD (Visible-Text Drafting), a speculative decoding method that uses OCR-recovered text as a retrieval source for drafting. We run OCR once, index the recognized text in a suffix automaton, and fuse its draft tree with a tree from a dynamic automaton over the prompt and accepted output. VTD requires no drafter training, external text corpus, or changes to target model weights. On five benchmarks, VTD achieves mean end-to-end speedups of $1.84\times$ on Qwen2.5-VL-3B, $1.97\times$ on Qwen2.5-VL-7B, and $1.62\times$ on InternVL2.5-8B. VTD outperforms every evaluated training-free retrieval baseline in speedup on every benchmark and target, and improves average speedup over the trained ViSpec drafter by 36.8% on Qwen2.5-VL-7B. These results show that visible text is a complementary source for retrieval drafting.

1 INTRODUCTION

Vision-language models (VLMs) (Liu et al., 2023; 2024; Li et al., 2023; Bai et al., 2025b) are increasingly used to transcribe images into structured text: a screenshot into HTML, a scanned page into Markdown, or a table into a machine-readable representation. These outputs often contain hundreds or thousands of tokens, so inference time is dominated by the autoregressive decode loop rather than by the one-time image encoding step. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) addresses this bottleneck with a cheap drafter that proposes several tokens and a target model that verifies them in one forward pass, without changing the target's output. Its practical benefit, however, depends on how often the drafter proposes the target's next tokens.

Existing VLM drafters expose a fundamental trade-off. *Trained* drafters condition on visual or hidden representations and can achieve high acceptance rates (Lin et al., 2025; Kang et al., 2026; Hu et al., 2026; Xie et al., 2026; Lee et al., 2026; Zhu et al., 2026), but they require a target-specific training procedure and are unavailable for many released checkpoints. *Retrieval* drafters require no training: they copy continuations from the prompt (Somasundaram et al., 2025), an external corpus (He et al., 2024), previous requests (Oliaro et al., 2026), or the current generation (Hu et al., 2025). These methods are limited by the text they index. Before a token is generated, text printed in the image is absent from the token history, even when it determines the next part of the response.

We first diagnose this limitation on a subset of Design2Code, a benchmark for generating HTML from webpage screenshots (Si et al., 2025), using Qwen2.5-VL-7B (Bai et al., 2025b). For a dynamic suffix automaton over the prompt and generated sequence, mean accepted length rises from 1.56 in the first three deciles to 2.15 in the last three, as repeated markup structure accumulates. Nevertheless, 66.1% of decode steps accept no draft descendant. At these *miss* steps, only 2.3% of the three-token spans begin at an earlier position in the same output, limiting what a different branch choice could recover from that index. Tokens classified as words, numbers, or color literals account for 56.0% of misses. These categories are not exclusively image text, but their concentration

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.

suggests that many residual misses depend on content that must be read from the image rather than retrieved from the generated history.

This observation suggests a simple complementary source. Instead of training a drafter to interpret the full visual representation, we can expose the text that is legible in the image through an off-the-shelf optical character recognition (OCR) model (JaidedAI, 2020). OCR is not expected to recover every target token, but it can make image-specific words, numbers, and literals available to a retrieval drafter without target-specific training or an external corpus.

We introduce **VTD** (Visible-Text Drafting), a training-free speculative decoding method for VLMs. Before decoding, VTD applies OCR once to the input image, tokenizes the recognized text with the target's tokenizer, and builds a suffix automaton over those tokens. During decoding, a second suffix automaton indexes the prompt and accepted output. Each source proposes a tree rooted at the target's next greedy token; VTD merges the two trees and verifies their union in one target forward pass. The sources can therefore contribute different branches at the same step.

Across five image-to-markup benchmarks and three target configurations spanning two VLM families, VTD achieves the highest average speedup for every target: $1.84\times$ on Qwen2.5-VL-3B, $1.97\times$ on Qwen2.5-VL-7B, and $1.62\times$ on InternVL2.5-8B. It also outperforms the strongest training-free baseline on every benchmark and improves average speedup over the trained ViSpec drafter by 36.8% on Qwen2.5-VL-7B. These results support visible text as a complementary retrieval source for long-form visual transcription. We summarize our contributions as follows:

- We analyze retrieval drafting for visual transcription and identify the gap between image-determined content and what token-history indices can retrieve.
- We introduce VTD, which runs a single off-the-shelf OCR pass, indexes the recognized per-image text, and fuses it with online prompt/output retrieval in one draft tree. To best of our knowledge, VTD is the first to use OCR-recovered visible text as a draft source.
- We evaluate VTD across five benchmarks and three target configurations spanning two VLM families. It achieves the highest average speedup for every target and, a statistically significant lead in average speedup on every target.

2 Related Work

2.1 Speculative Decoding

Speculative decoding accelerates autoregressive inference by having an inexpensive drafter propose candidate tokens and the target model verify them in parallel, while preserving the target distribution Leviathan et al. (2023). Existing methods differ primarily in how they obtain the draft.

Model-based drafters. A dominant line trains a drafter. Medusa Cai et al. (2024) attaches multiple decoding heads to the target model, whereas EAGLE and its successors Li et al. (2024a;b; 2026) train shallow autoregressive heads on the target model's hidden features. These approaches can achieve strong drafter and target alignment, but require additional training and are generally coupled to a particular target checkpoint.

Retrieval-based drafters. A complementary, training-free approach retrieves candidate continuations from text observed before decoding. Prompt Lookup Decoding (PLD) Somasundaram et al. (2025) copies tokens that follow occurrences of the current suffix in the input prompt, and is therefore most effective when the output reuses input spans. REST He et al. (2024) retrieves continuations from a static corpus with a suffix array, while CREST Ho et al. (2024) reduces the storage cost of this datastore. SuffixDecoding (SuffixD) Oliaro et al. (2026) mines recurring patterns in a model's own previous outputs, including repetitive agentic trajectories. SAM Decoding (SAM-D) Hu et al. (2025) generalizes exact n gram lookup with a suffix automaton that indexes both a static corpus and the current generation. Because the target verifies every proposed token, retrieval-based methods are plug and play and preserve the target output under the usual exact-logit assumption. Their attainable speedup is nevertheless limited by output *retrievability*, that is, the frequency with which a useful continuation appears in the available text. We extend suffix automaton retrieval to multimodal code generation VLMs and show that retrievability remains the principal determinant of speedup.

2.2 Speculative Decoding for Vision Language Models

Multimodal drafters. Work on speculative decoding for VLMs spans trained drafters that consume visual representations and training-free external proposals. Because exposing a drafter to the full visual representation can reduce acceptance, several methods compress or distill visual information before drafting Gagrani et al. (2024); Kang et al. (2026); Xie et al. (2026); Hu et al. (2026); GLANCE Lee et al. (2026) drafts a block in one pass from the target's fused state, FOVEA Zhu et al. (2026) retrieves projected patch features per image, and HSD Liao et al. (2026) pairs an external drafter with detected document regions. All draw drafts from visual features or layout. We instead exploit the literal text visible in the image: an off-the-shelf OCR model runs once, its output is indexed by a suffix automaton, and this per-image source is fused with an online automaton over the generated sequence—no drafter training, target-specific parameters, or region detection.

Acceptance dynamics. Existing analyses usually report an aggregate acceptance rate. Finer grained studies consider acceptance by draft position, where later positions may suffer from feature drift Huang et al. (2025), or by input context length Cha et al. (2026); Lee et al. (2025). We study acceptance along the generated sequence itself and identify within sequence recurrence: for structured outputs, accepted length increases as the online index accumulates recurring structure.

3 Background and Motivation

3.1 Preliminaries

Speculative decoding. An autoregressive model with distribution p emits one token per step, $y_t \sim p(\cdot \mid x, y_{<t})$, each requiring a target forward pass. Since verifying several candidate tokens in one pass costs little more than verifying one, speculative decoding has a cheap *drafter* propose γ candidates $\tilde{y}_{t:t+\gamma-1}$ that the target verifies in a single parallel pass. Under greedy decoding the verifier accepts the longest correct prefix and appends its own token at the first mismatch, so every step yields at least one token and more when the draft is accurate. With identical conditional logits and deterministic tie-breaking, the emitted sequence is identical to standard greedy decoding. We call the tokens produced per step the *accept length*: speedup comes from amortizing one forward pass over several accepted tokens. Arranging multiple candidate continuations in a token tree, verified jointly under a tree-structured attention mask, raises it further.

Vision-language model decoding. A VLM encodes the input image into visual tokens, prepends them to the prompt tokens, and decodes the response autoregressively. Inference thus splits into a parallel *prefill* over image and prompt and a sequential *decode* loop. Visual tokens make the prefix long and prefill costly, but this is paid once, whereas decode pays a forward pass per token. Our outputs are structured HTML or Markdown transcriptions often running to hundreds or thousands of tokens, so decode dominates end-to-end latency and is the target for acceleration.

Suffix automaton retrieval. Our method builds on SAM-D Hu et al. (2025), whose suffix-automaton index tracks, in amortized $\mathcal{O}(1)$ time per token, the longest suffix of the current output that recurs in the index and emits the following tokens as the draft. The design maintains two such indices in parallel, distinguished by what they index. A *static* index is built offline over an external corpus and supplies continuations that recur *across* examples, such as boilerplate shared between documents. A *dynamic* index is built online over the tokens generated so far and grown with each accepted token, supplying continuations that recur *within* the current output. This within-output index is central to what follows.

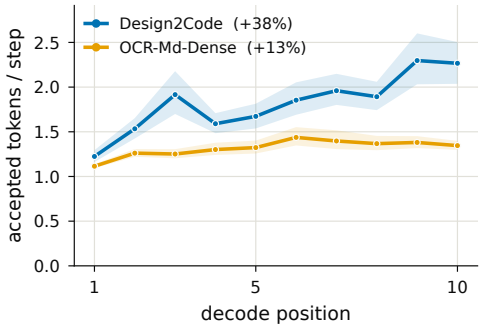
3.2 Key Observations

We study dynamic retrieval with Qwen2.5-VL-7B through two questions: how acceptance length varies over the course of generation and how often continuations at steps with no accepted draft descendants recur earlier in the same output.

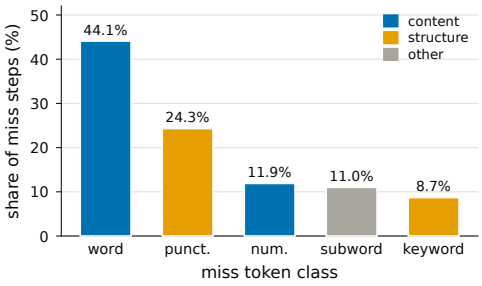
Observation 1: acceptance rises along the generated sequence. The dynamic automaton indexes the prompt and accepted output. As decoding proceeds, the accepted output provides addi-

tional material for retrieval, including recurring tags, attributes, and formatting patterns. For each page, we normalize decoding positions by the page's total number of decoding steps, partition the steps into ten deciles, and compute the mean acceptance length within each decile. We then average across pages and compare the first and last three deciles. Mean acceptance length increases from 1.56 to 2.15 on Design2Code and from 1.21 to 1.36 on OCR-Md-Dense (Figure 1a), corresponding to relative increases of approximately 38% and 13%, respectively.

These aggregate trends are consistent with increasing reuse of output structure as the generated history grows. The larger increase on Design2Code also suggests that the usefulness of dynamic retrieval depends on output structure. The trend is not monotonic on every page, so acceptance by decoding position provides information that is hidden by a single overall average.



(a) Acceptance rises along the generated sequence.



(b) Token classes at miss steps.

Figure 1: The two observations that motivate VTD. **(a)** Dynamic-only mean accepted length by decode position, with 95% bootstrap intervals over pages. **(b)** Single-token *miss* steps on Design2Code by the emitted token's heuristic class; colors distinguish content, structure, and other tokens.

Observation 2: residual misses show limited sequence recurrence. On Design2Code, 7,269 of 10,995 dynamic-only decoding steps (66.1%) have acceptance length one: the target-selected root token is emitted, but no draft descendant is accepted. We call these steps *misses*. For each miss, we classify the emitted token using lexical heuristics. We then retrospectively examine the three-token span beginning at that position and check whether the identical span begins at an earlier position in the completed generated output. Overlapping occurrences are allowed. This test measures recurrence in the output and is an oracle-style diagnostic; it does not describe what the online index knew at the time of the miss.

Only 2.3% of misses satisfy this recurrence criterion. Thus, exact three-token recurrence provides little additional headroom for a retrieval policy restricted to previously generated spans. The content classes *word* and *number/color* account for 44.1% and 11.9% of misses, respectively, for a combined 56.0%. Their recurrence rates are only 0.2% and 3.2%. These classes may contain image-specific text, but they also include code literals that are not directly readable from the image. The breakdown therefore motivates evaluating visible text as an additional retrieval source; it does not establish what fraction of misses OCR can resolve.

Implication. Dynamic retrieval becomes more useful as recurring output structure accumulates, but many miss positions do not have an identical three-token continuation earlier in the generated output. This pattern motivates a second, request-local source that can expose text present in the input image before it appears in the response. We therefore complement sequence-history retrieval with an OCR-derived visible-text index.

4 Methodology

We present VTD (Visible-Text Drafting), a training-free speculative decoding method that combines two request-local retrieval sources: a dynamic index over the prompt and accepted output, and a fixed per-image index over OCR-recognized text. At each decoding iteration, both sources propose candidate continuations under the same target-selected root. VTD merges the source trees, verifies the merged tree in one target forward pass, and updates both retrieval states using only the accepted tokens. Figure 2 gives the complete procedure.

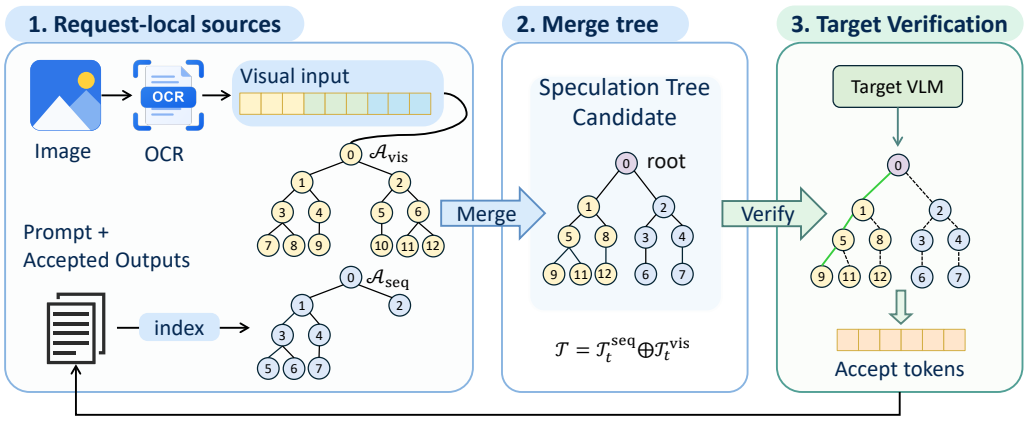


Figure 2: **Overview of VTD. (1) Request-local sources.** A single OCR pass over the input image produces text that is tokenized with the target tokenizer and stored in the fixed per-image index $\mathcal{A}_{\text{vis}}$. The online index $\mathcal{A}_{\text{seq}}$ stores the prompt and accepted output tokens and is extended after each accepted token. Both sources construct a candidate tree for the current target-selected root. **(2) Tree merging.** The two roots are identified and the remaining nodes are combined into one tree, preserving distinct branches from both sources. **(3) Target verification.** The target evaluates all nodes of the merged tree in one forward pass with a tree-attention mask. VTD accepts the longest argmax-consistent path, emits its tokens, updates the online retrieval state, advances the OCR match state, and uses the next target argmax token as the root of the following iteration.

4.1 Two Corpus-Free Retrieval Sources

Let I denote the input image, x the text prompt, and $y_{<t}$ the accepted output prefix before decoding iteration t. VTD uses no external corpus or cross-request database during normal decoding. It maintains two request-local retrieval indexes: (i) a dynamic suffix automaton $\mathcal{A}_{\text{seq}}$ over the prompt and accepted output, updated online; and (ii) a fixed per-image suffix automaton $\mathcal{A}_{\text{vis}}$ over the OCR tokens obtained from I. The external static corpus is used only in the explicit source ablation.

In-sequence retrieval. The first source is the dynamic suffix automaton introduced by SAM-D (Hu et al., 2025). It is initialized from the prompt and extended after each accepted output token. At each iteration, it retrieves continuations that recur within the current request. Because the indexed sequence grows during decoding, this source becomes increasingly useful when the output reuses markup patterns such as tags, attributes, and formatting scaffolding.

Visible-text retrieval. The second source is the contribution of this work. We apply an OCR model Φ once to obtain a text string $\Phi(I)$, tokenize it with the target model's tokenizer, and construct the fixed per-image suffix automaton $\mathcal{A}_{\text{vis}}$. The index is available before generation starts and is not modified during decoding. Its match state is advanced using the accepted output tokens, allowing it to track the current context while retaining the original OCR text. It can therefore propose continuations that occur in the image even when they have not appeared in the generated prefix.

The two sources provide complementary evidence. The dynamic index retrieves recurring output structure and becomes more useful as the response grows, whereas the visible-text index exposes text recovered directly from the current image. Because the visible index is constructed independently for each request, it requires neither target-specific training nor a domain-specific text corpus.

4.2 Fusing Two Sources into One Verified Tree

Motivation for fusion. At each step, one could select the source with the longer match, as in a single-source retrieval drafter. Selection discards branches proposed by the other source, even when the two sources contain different correct continuations. VTD instead takes their union under a shared root. This increases the number of candidates presented to the target, but keeps verification in one

tree-structured forward pass. The maximum tree size is bounded by the per-source budget, and the resulting verification overhead is measured in Appendix B.2.

Per-source drafting. Let $\tilde{y}_t$ be the target model's next greedy token from the previous verification pass. It serves as the root of the current draft tree and remains pending until that tree is verified. For source $s \in \{\text{seq}, \text{vis}\}$, let (i_s, ℓ_s) be the state and match length obtained by tentatively extending the source with $\tilde{y}_t$. Equivalently, the matched suffix is the longest suffix of $x \circ y_{<t} \circ \tilde{y}_t$ represented by the source. We allocate

$$n_s = \min(N,\ 1 + \lfloor \alpha\ \max(0, \ell_s - b_s) \rfloor) \tag{1}$$

nodes to source s. The shared root accounts for one node, so source s contributes at most $n_s - 1$ descendants. The dynamic source expands its tree breadth first. The visible source ranks outgoing transitions by cumulative transition frequency, considers its top-K transitions at each state, and admits at most K nodes per depth. Thus, N bounds the total number of nodes from each source, while K controls the visible source's per-depth branching.

The offset b_s discounts short matches. In the reported fusion configuration, $b_{\text{seq}} = b_{\text{vis}} = 0$: the dynamic and visible indices are small, request-specific sources for which short matches can still be useful. The external static-corpus baseline uses $b_{\text{static}} = 5$, reflecting its larger corpus and greater risk of spurious short matches. This distinction is important because applying the corpus-scale offset to a single OCR page would remove most of the visible source's candidates.

Tree merging and verification. Both source trees begin with $\tilde{y}_t$. We retain one copy of this shared root, append the non-root nodes from the visible tree to the dynamic tree, and remap their parent indices. The merged tree is

$$\mathcal{T}_t = \mathcal{T}_t^{\text{seq}} \oplus \mathcal{T}_t^{\text{vis}} \tag{2}$$

with at most

$$n_{\text{seq}} + n_{\text{vis}} - 1 \leq 2N - 1 \tag{3}$$

nodes. Its parent array defines a tree-attention mask in which each candidate attends to the cached multimodal context, itself, and its ancestors. The target scores all branches in one forward pass and accepts the longest root-to-node path whose tokens agree with the target's conditional argmax. The target's argmax token after the accepted path becomes $\tilde{y}_{t+1}$, the pending root for the next iteration.

The target's acceptance rule is unchanged by fusion. If tree and autoregressive evaluation produce identical conditional logits and use deterministic tie breaking, induction over decoding steps gives the same output as greedy autoregressive decoding. An incorrect OCR continuation is rejected by the target.

4.3 Cost Model

VTD adds one OCR inference pass and per-iteration retrieval, tree construction, and verification overhead. The OCR model runs once before decoding; its median runtime is 0.75 seconds across the five benchmarks, compared with median online generation times of 7.8–13.9 seconds across targets. The reported end-to-end interval includes visual prefill, OCR inference, retrieval, tree construction, target verification, and cache updates. Tokenization and suffix-automaton construction are performed once per request and excluded from this interval. Possible overlap between OCR and visual prefill is analyzed separately in Appendix B.2 and is not used in the main results.

Let $M = n_{\text{seq}} + n_{\text{vis}} - 1 \leq 2N - 1$ denote the merged tree size. Each decoding iteration performs one amortized $\mathcal{O}(1)$ suffix-state lookup per source, constructs the merged tree, and runs one target forward pass over its M nodes. For fixed K, tree construction is approximately $\mathcal{O}(M \log M)$ because candidate branches are prioritized with a heap; merging is $\mathcal{O}(M)$, and mask construction is $\mathcal{O}(M^2)$ in the worst case.

5 Experiments

In this section, we evaluate VTD across diverse VLMs and benchmarks. We detail the experimental setup in §5.1, present the main comparative results in §5.2, conduct extensive ablation studies in §5.3 to validate each component, more results are presented in Appendix B.

5.1 Experimental Settings

Benchmarks. We evaluate our method on five multimodal generation benchmarks covering two task families and a broad range of visual text densities. Three benchmarks focus on generating HTML from webpage screenshots, namely Design2Code (Si et al., 2025), WebCode2M (Gui et al., 2025), and OCR-Md-Dense (PrithivMLmods, 2024). The remaining two benchmarks evaluate document parsing by requiring models to transcribe document pages into Markdown, namely OmniDocBench (Ouyang et al., 2025) and the OmniAI OCR Benchmark (OmniAI-OCR) (Omni AI, 2024). Collectively, these benchmarks range from visually structured webpages to dense, text-heavy document pages, allowing us to assess whether the observed acceleration generalizes across markup formats and document text densities. For each dataset, we evaluate a fixed-seed random subset of 100 examples. Additional dataset details are provided in Appendix A.1.

Compared baselines. We compare against two groups of speculative-decoding methods. The first are *training-free* retrieval-based drafters: PLD Somasundaram et al. (2025), REST He et al. (2024), SuffixD Oliaro et al. (2026), and SAM-D (dynamic-only) Hu et al. (2025). For SAM-D, we disable the external corpus and use only the dynamic index over the prompt and accepted output; the external-corpus configuration appears only in the source ablation. The second is a *training-based* reference, ViSpec Kang et al. (2026), which uses a learned, vision-aware drafter to generate candidate continuations conditioned on the input image. In our evaluation, we use the publicly released ViSpec checkpoint corresponding to Qwen2.5-VL-7B. All methods are implemented or adapted following their official releases, and detailed configuration choices are provided in Appendix A.3.

Implementation details. Our main experiments evaluate three VLMs across two model families: Qwen2.5-VL-3B-Instruct, Qwen2.5-VL-7B-Instruct (Bai et al., 2025b), and InternVL2.5-8B (Chen et al., 2024). More results are provided in Appendix B. The OCR drafter is powered by EasyOCR. All methods run under greedy decoding with a batch size of one and a $6,144$-token generation cap. Since speculative decoding is lossless, it produces the same output as autoregressive decoding on a given input; we therefore report speedup only on examples that terminate naturally, i.e., emit an end-of-sequence token before reaching the generation cap (the *natural-EOS* filter). Truncated generations that hit the cap are excluded, as their wall-clock time reflects the cap rather than the model's true stopping behavior and would otherwise distort the speedup measurement; the same filter is applied identically to every method and target. See Appendix A.3 for additional hyperparameters.

Metrics. We report two evaluation metrics: **(1) Mean Accepted Tokens (MAT)** is the average number of tokens produced per target verification pass. MAT measures how effectively target-model forward-pass work is amortized, but does not account for OCR, drafting, or other overheads; and **(2) Speedup**, which measures the wall-clock acceleration over standard autoregressive decoding for the reported inference pipeline. Formal definitions are given in Appendix A.2.

5.2 Main Results

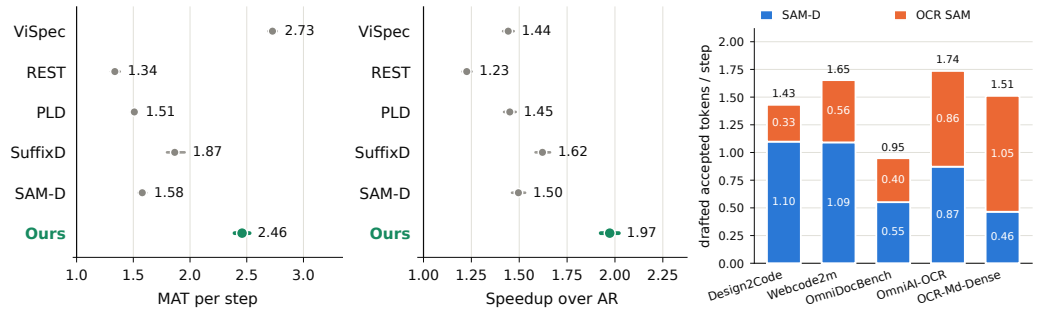
Table 1 reports MAT and speedup across five benchmarks and three target configurations. VTD achieves the highest average speedup for each target, reaching $1.84\times$ on Qwen2.5-VL-3B, $1.97\times$ on Qwen2.5-VL-7B, and $1.62\times$ on InternVL2.5-8B. These averages use the common natural-EOS subset for the training-free methods; ViSpec uses its own valid subset because it is evaluated with a separate harness. The confidence intervals and source attribution analysis below examine the statistical stability and origin of these aggregate gains.

Against training-free retrieval. VTD attains the highest speedup on every benchmark and target in the table. Relative to SuffixD, the strongest training-free baseline by average speedup for each target, the average improvement is 21.1% on Qwen2.5-VL-3B, 21.6% on Qwen2.5-VL-7B, and 14.9% on InternVL2.5-8B. The gains are especially clear on text dense inputs. On Qwen2.5-VL-7B, VTD reaches $1.78\times$ and $2.17\times$ on OCR-Md-Dense and OmniAI-OCR, compared with $1.31\times$ and $1.90\times$ for the strongest baseline on those datasets. It is also the only method exceeding $1.5\times$ on OmniDocBench for this target. This consistent advantage across both model families supports the complementarity of image-grounded retrieval and generated sequence history.

The comparison with SAM-D isolates the contribution of the visible text source. Averaged across the five benchmarks, VTD increases MAT from 1.58 to 2.46 and speedup from $1.50\times$ to $1.97\times$.

Table 1: Results on WebCode2M, Design2Code, OmniDocBench, OCR-Md-Dense, OmniAI-OCR, with Qwen2.5-VL-3B, Qwen2.5-VL-7B and InternVL2.5-8B target models. Reported in MAT and speedup. † denotes training-based methods. Best results are bold.

Model	Method	WebCode2M		Design2Code		OmniDocBench		OCR-Md-Dense		OmniAI-OCR		Average	
		MAT	Speedup	MAT	Speedup	MAT	Speedup	MAT	Speedup	MAT	Speedup	MAT	Speedup
Qwen2.5-VL-3B	AR	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×
	REST	1.21	1.10×	1.34	1.22×	1.14	1.05×	1.37	1.19×	1.63	1.47×	1.34	1.21×
	PLD	1.75	1.58×	1.72	1.68×	1.33	1.25×	1.29	1.20×	1.51	1.38×	1.52	1.42×
	SuffixD	2.31	1.66×	2.06	1.81×	1.35	1.19×	1.55	1.20×	2.19	1.73×	1.89	1.52×
	SAM-D	1.99	1.61×	1.83	1.73×	1.38	1.28×	1.30	1.18×	1.62	1.44×	1.62	1.45×
	VTD	**2.90**	**2.17**×	**2.29**	**2.01**×	**2.03**	**1.42**×	**2.66**	**1.61**×	**2.81**	**2.00**×	**2.54**	**1.84**×
Qwen2.5-VL-7B	AR	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×
	ViSpec†	**2.69**	1.41×	**2.95**	1.57×	**2.46**	1.30×	2.44	1.28×	**3.12**	1.66×	**2.73**	1.44×
	REST	1.21	1.12×	1.32	1.20×	1.15	1.10×	1.37	1.21×	1.64	1.50×	1.34	1.23×
	PLD	1.65	1.56×	1.71	1.68×	1.32	1.29×	1.30	1.24×	1.57	1.48×	1.51	1.45×
	SuffixD	2.01	1.69×	2.15	1.90×	1.37	1.32×	1.57	1.31×	2.24	1.90×	1.87	1.62×
	SAM-D	1.73	1.60×	1.82	1.75×	1.37	1.32×	1.32	1.25×	1.67	1.56×	1.58	1.50×
	VTD	2.54	**2.22**×	2.37	**2.14**×	2.08	**1.56**×	**2.60**	**1.78**×	2.70	**2.17**×	2.46	**1.97**×
InternVL2.5-8B	AR	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×
	REST	1.13	0.90×	1.24	1.00×	1.09	0.88×	1.13	0.89×	1.44	1.13×	1.21	0.96×
	PLD	1.73	1.38×	1.70	1.45×	1.30	1.12×	1.27	1.05×	1.55	1.29×	1.51	1.26×
	SuffixD	2.17	1.60×	2.38	1.79×	1.27	1.05×	1.41	1.09×	1.90	1.46×	1.83	1.40×
	SAM-D	1.80	1.44×	1.97	1.55×	1.34	1.15×	1.29	1.06×	1.59	1.32×	1.60	1.30×
	VTD	**2.45**	**1.85**×	**2.42**	**1.87**×	**1.91**	**1.26**×	**2.45**	**1.53**×	**2.21**	**1.58**×	**2.29**	**1.62**×



(a) Average MAT (95% CI). (b) Average speedup (95% CI). (c) Drafted tokens by source.

Figure 3: VTD and the baselines on Qwen2.5-VL-7B. Panels **(a)** and **(b)** show mean MAT and end-to-end speedup averaged over five benchmarks, with 95% page-bootstrap confidence intervals. Panel **(c)** decomposes accepted tokens per step into the verifier's one-token bonus, the in-sequence dynamic index (blue), and the per-image visible-text index (orange).

This gain requires no additional training or static text corpus, although the OCR pass and its latency are included in all reported end-to-end measurements. SuffixD is comparatively competitive on homogeneous HTML layouts such as Design2Code, where cross request history can capture recurring markup. Its advantage is smaller on heterogeneous document pages, where the visible text varies from image to image. The result indicates that the additional source supplies information that generated token history alone cannot reliably recover.

Against trained drafters. On Qwen2.5-VL-7B, VTD improves average speedup over the trained ViSpec baseline from 1.44× to 1.97×, a relative gain of 36.8%. ViSpec has higher average MAT (2.73 versus 2.46), but it generates those drafts with a learned auxiliary network. VTD instead uses an OCR-based visible text index, and the associated OCR cost is included in the end-to-end timing. The results therefore illustrate that MAT alone does not determine wall-clock performance. ViSpec also requires a target-specific trained checkpoint and is evaluated here only for Qwen2.5-VL-7B, whereas VTD runs without target-specific training on InternVL2.5-8B and achieves 1.62× average speedup. VTD and ViSpec therefore use complementary draft sources: ViSpec derives candidates from model states, whereas VTD retrieves text observed in the image. This portability is useful when a target-specific draft checkpoint is unavailable or expensive to train.

Statistical robustness and source attribution. Figure 3 summarizes MAT and speedup with confidence intervals and attributes accepted tokens to their source on Qwen2.5-VL-7B. Panels 3a and 3b show point estimates with 95% bootstrap intervals obtained by resampling pages. VTD has the highest speedup, and its interval does not overlap that of any training free baseline, including Suf-

Table 2: Source ablation. The dynamic-only row uses the prompt/accepted-output index without an external corpus. The + static corpus row adds an external held-out cross-page corpus and is included only as an ablation; it is not part of VTD. The + visible text (VTD) row adds the fixed per-image OCR index to the dynamic-only configuration. We report MAT, speedup, and relative speedup gain over dynamic-only. "–" denotes an unavailable static-corpus run.

Configuration	WebCode2M			Design2Code			OmniDocBench			OCR-Md-Dense			OmniAI-OCR		
	MAT	Speedup	Δ	MAT	Speedup	Δ	MAT	Speedup	Δ	MAT	Speedup	Δ	MAT	Speedup	Δ
SAM-D	1.71	1.59×	–	1.82	1.74×	–	1.36	1.31×	–	1.32	1.25×	–	1.67	1.56×	–
+ static corpus	2.09	1.71×	+7.5%	1.90	1.79×	+2.8%	1.41	1.40×	+6.7%	–	–	–	1.86	1.76×	+12.8%
+ visible text (VTD)	2.52	2.22×	+39.6%	2.37	2.14×	+23.0%	2.03	1.52×	+16.0%	2.60	1.78×	+42.4%	2.70	2.17×	+39.1%

Table 3: OCR engine ablation. We report MAT and speedup, also give the relative speedup gain over PaddleOCR. PaddleOCR reads more of the page and raises MAT on nearly every dataset, yet loses wall-clock *everywhere*. The operative criterion for the drafter is therefore accuracy *per unit of latency*, not accuracy; VTD uses EasyOCR.

Engine	WebCode2M			Design2Code			OmniDocBench			OCR-Md-Dense			OmniAI-OCR		
	MAT	Speedup	Δ	MAT	Speedup	Δ	MAT	Speedup	Δ	MAT	Speedup	Δ	MAT	Speedup	Δ
PaddleOCR	3.35	1.71×	–	2.52	1.62×	–	2.65	1.06×	–	3.51	0.97×	–	2.94	1.29×	–
EasyOCR	2.52	2.22×	+29.8%	2.37	2.13×	+31.5%	2.03	1.52×	+43.4%	2.60	1.78×	+72.7%	2.70	2.17×	+68.2%

fixD at $1.62\times$ [1.59, 1.66]. ViSpec has higher MAT than VTD (2.73 versus 2.46), but its learned drafter adds computation on the critical path, so its wall clock speedup is lower. Panel 3c shows that the visible text index supplies between 34% of drafted tokens on webpage screenshots and 69% on dense document pages. Its contribution therefore grows with the amount of text present in the image, as predicted by our observations in §3.2.

5.3 Ablation Studies

Per-image text is more useful than a cross-page corpus. Table 2 isolates the effect of the second retrieval source on top of the dynamic index. On the four datasets with an available static-corpus ablation, the cross-page corpus improves end-to-end speedup by only 2.8–12.8%, whereas the per-image visible-text index improves it by 16.0–42.4% on the same four datasets. The visible index also gives the largest gains on the document benchmarks, including a near doubling of MAT on OCR-Md-Dense ($1.32 \rightarrow 2.60$) and a 42.4% speedup gain. The comparison supports the role of image-specific content in the diagnostic of §3.2: expanding text retrieval across examples adds little when the relevant text is unique to the current image, while indexing that image's text supplies a complementary source. This result motivates the corpus-free design of VTD and avoids a domain-specific static datastore.

Higher OCR acceptance does not guarantee faster decoding. Table 3 compares the default EasyOCR reader with the stronger PaddleOCR (PP-OCRv5) (Cui et al., 2026). PaddleOCR produces higher MAT on all five benchmarks, including a 35.0% increase on OCR-Md-Dense, but speedup decreases on every benchmark. The contrast is clearest on OCR-Md-Dense: MAT increases from 2.60 to 3.51, while speedup drops from $1.78\times$ to $0.97\times$, turning the method into a net slowdown. Because the OCR pass is on the critical path, PaddleOCR's additional latency outweighs the decoding work saved by its more accurate drafts. OCR selection should therefore optimize the accuracy–latency trade-off rather than acceptance or MAT alone.

6 Conclusion

We introduced VTD, a speculative decoding method for VLMs that retrieves draft tokens from text visible in the input image. One OCR pass builds a per-image text index. During decoding, VTD combines candidates from this index with candidates retrieved from the prompt and generated sequence, then verifies them jointly with the target. It requires no drafter training, external text corpus, or changes to the target weights. Across five image-to-markup benchmarks and three target configurations, VTD achieves the highest average end-to-end speedup among the evaluated methods. The ablations show that visible text complements sequence history, but the cost of OCR can offset gains from higher draft acceptance. The benefit therefore depends on the amount and relevance of image text, OCR language coverage, and extraction latency. Future work could adapt OCR effort to each input and combine visible-text retrieval with learned drafters.

AI USE STATEMENT

In this work, we used generative AI tools solely for language editing, including improving grammar, clarity, readability, and presentation. All AI-assisted revisions were carefully reviewed by the authors to ensure that they accurately reflected the intended technical meaning and did not introduce unsupported claims or incorrect information.

ETHICS STATEMENT

This work uses publicly released benchmark datasets and pretrained models; we do not collect new human data, interact with participants, or infer sensitive attributes. Some benchmark images may contain personal-looking or sensitive content, including forms, identity documents, and contracts. We use these datasets only for aggregate model evaluation, follow the providers' terms of use, and do not attempt to identify individuals or release source images or reconstructed personal text. Qualitative examples in the appendix are selected and, where necessary, redacted to remove names, addresses, account numbers, signatures, and other direct identifiers.

REPRODUCIBILITY STATEMENT

We describe the proposed method, model configurations, datasets, and evaluation metrics in detail in the main paper and appendix. Upon acceptance, we will release the complete implementation and experiment scripts for reproducing our results.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

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, 2025b. 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. *arXiv preprint arXiv:2401.10774*, 2024.

Jungyoub Cha, Hyunjong Kim, and Sungzoon Cho. Specextend: A drop-in enhancement for speculative decoding of long sequences. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 43366–43377, 2026.

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.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024.

Cheng Cui, Yubo Zhang, Ting Sun, Xueqing Wang, Hongen Liu, Manhui Lin, Yue Zhang, Tingquan Gao, Changda Zhou, Jiaxuan Liu, Zelun Zhang, Jing Zhang, Jun Zhang, and Yi Liu. Pp-ocrv5: A specialized 5m-parameter model rivaling billion-parameter vision-language models on ocr tasks, 2026. URL https://arxiv.org/abs/2603.24373.

Mukul Gagrani, Raghavv Goel, Wonseok Jeon, Junyoung Park, Mingu Lee, and Christopher Lott. On speculative decoding for multimodal large language models. *arXiv preprint arXiv:2404.08856*, 2024.

Yi Gui, Zhen Li, Yao Wan, Yemin Shi, Hongyu Zhang, Bohua Chen, Yi Su, Dongping Chen, Siyuan Wu, Xing Zhou, et al. Webcode2m: A real-world dataset for code generation from webpage designs. In *Proceedings of the ACM on Web Conference 2025*, pp. 1834–1845, 2025.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. Rest: Retrieval-based speculative decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1582–1595, 2024.

Sophia Ho, Jinsol Park, and Patrick Wang. Crest: Effectively compacting a datastore for retrieval-based speculative decoding. *arXiv preprint arXiv:2408.04678*, 2024.

Yunhai Hu, Tianhua Xia, Zining Liu, Rahul Raman, Xingyu Liu, Bo Bao, Eric Sather, Vithursan Thangarasa, and Sai Qian Zhang. Dream: Drafting with refined target features and entropy-adaptive cross-attention fusion for multimodal speculative decoding. *Advances in Neural Information Processing Systems*, 38:167592–167612, 2026.

Yuxuan Hu, Ke Wang, Xiaokang Zhang, Fanjin Zhang, Cuiping Li, Hong Chen, and Jing Zhang. Sam decoding: Speculative decoding via suffix automaton. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12187–12204, 2025.

Langlin Huang, Chengsong Huang, Jixuan Leng, Di Huang, and Jiaxin Huang. Poss: Position specialist generates better draft for speculative decoding. *arXiv preprint arXiv:2506.03566*, 2025.

JaidedAI. EasyOCR: Ready-to-use optical character recognition package. https://github.com/JaidedAI/EasyOCR, 2020. Accessed: 2026-09-20.

Jialiang Kang, Han Shu, Wenshuo Li, Yingjie Zhai, and Xinghao Chen. Vispec: Accelerating vision-language models with vision-aware speculative decoding. *Advances in Neural Information Processing Systems*, 38:115511–115532, 2026.

Jaeseong Lee, Aurick Qiao, Gabriele Oliaro, Ye Wang, Samyam Rajbhandari, et al. Owl: Overcoming window length-dependence in speculative decoding for long-context inputs. *arXiv preprint arXiv:2510.07535*, 2025.

Jungseob Lee, Seongtae Hong, Dongyub Jude Lee, Chanjun Park, Jaehyung Seo, Sugyeong Eo, and Heuiseok Lim. Vision is not overhead: One-pass block drafting for lossless speculative decoding in vision-language models, 2026. URL https://arxiv.org/abs/2609.00355.

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

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023. URL https://arxiv.org/abs/2301.12597.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7421–7432, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *Advances in Neural Information Processing Systems*, 38:136737–136756, 2026.

Wenhui Liao, Hongliang Li, Pengyu Xie, Xinyu Cai, Yufan Shen, Yi Xin, Qi Qin, Shenglong Ye, Tianbin Li, Ming Hu, et al. Hsd: Training-free acceleration for document parsing vision-language models with hierarchical speculative decoding. *arXiv preprint arXiv:2602.12957*, 2026.

Luxi Lin, Zhihang Lin, Zhanpeng Zeng, and Rongrong Ji. Speculative decoding reimagined for multimodal large language models. *arXiv preprint arXiv:2505.14260*, 2025.

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

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26286–26296. IEEE, 2024.

Gabriele Oliaro, Zhihao Jia, Daniel Campos, and Aurick Qiao. Suffixdecoding: Extreme speculative decoding for emerging ai applications. *Advances in Neural Information Processing Systems*, 38: 126326–126354, 2026.

Omni AI. Ocr benchmark: A comprehensive dataset for document parsing and ocr evaluation. https://huggingface.co/datasets/getomni-ai/ocr-benchmark, 2024. Accessed: 2026-08-22.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, et al. Omnidocbench: Benchmarking diverse pdf document parsing with comprehensive annotations. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24838–24848. IEEE, 2025.

PrithivMLmods. Ocr-markdown-dense-200x: A dataset for dense ocr to markdown conversion. https://huggingface.co/datasets/prithivMLmods/OCR-Markdown-Dense-200x, 2024. Accessed: 2026-08-22.

Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. Design2code: Benchmarking multimodal code generation for automated front-end engineering. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3956–3974, 2025.

Shwetha Somasundaram, Anirudh Phukan, and Apoorv Saxena. Pld+: Accelerating llm inference by leveraging language model artifacts. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 6090–6104, 2025.

Zhinan Xie, Peisong Wang, Shuang Qiu, and Jian Cheng. Hivis: Hiding visual tokens from the drafter for speculative decoding in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8952–8961, 2026.

Hengjie Zhu, Dayan Wu, Zihao Zhang, Xinze Liu, Jingxuan Yu, Peng Fu, Zheng Lin, Weiping Wang, and Ding Wang. Fovea: Focused on-demand visual evidence adaptation for cache-friendly multimodal speculative decoding, 2026. URL https://arxiv.org/abs/2608.22883.

APPENDIX

This appendix provides supplementary material that complements the main paper:

- Appendix A details the experimental setup, including the implementation and reproduction protocols for the compared baselines, as well as an estimate of computational overhead.
- Appendix B reports additional empirical analyses, including hyperparameter ablations and extended quantitative and qualitative results.
- Appendix C gives the complete decoding pseudocode.

A ADDITIONAL EXPERIMENTAL SETTINGS

A.1 EVALUATION BENCHMARKS

We evaluate our method on a diverse collection of multimodal generation and document parsing benchmarks. The benchmarks cover webpage-to-code generation, long-form document transcription, and structured OCR, allowing us to evaluate the method under different output lengths, visual layouts, and text densities.

Design2Code. Design2Code (Si et al., 2025) is a multimodal benchmark for evaluating whether vision-language models can translate webpage screenshots into their corresponding HTML and CSS implementations. It contains 484 real-world webpages sampled from the validation split of the C4 corpus, covering a broad range of webpage layouts, visual styles, and content structures.

WebCode2M. WebCode2M (Gui et al., 2025) is a large-scale dataset for generating webpage source code from visual designs. It contains approximately 2.56 million real-world image–code pairs, where each instance is associated with a webpage screenshot, its source code, and layout-related information. The examples are collected from publicly available web resources and filtered with a quality-scoring model to remove visually corrupted, incomplete, or low-quality webpages.

OmniDocBench. OmniDocBench (Ouyang et al., 2025) is a comprehensive benchmark for document parsing in realistic PDF scenarios. Its evaluation set contains 1,651 document pages spanning 10 document types, 5 layout categories, and 5 language categories, including academic papers, books, textbooks, exam papers, newspapers, magazines, research reports, presentations, handwritten notes, and historical documents. The benchmark provides rich annotations for document layout, reading order, text spans, tables, formulas, headers, footers, and other structural elements. It additionally includes text recognition, LaTeX formula annotations, and HTML table annotations, enabling evaluation of both end-to-end document transcription and individual parsing modules.

OCR-Markdown-Dense (OCR-Md-Dense). OCR-Markdown-Dense (PrithivMLmods, 2024) is a curated benchmark consisting of dense, high-resolution document pages paired with HTML transcription targets; 99% of the targets begin with an HTML tag. The pages contain substantial amounts of text together with tables, formulas, and complex layout structures, making the benchmark suitable for evaluating long-form visual transcription with HTML output.

Omni-OCR-Benchmark (OmniAI-OCR). OmniAI-OCR-Benchmark (Omni AI, 2024) is a multimodal OCR benchmark designed to evaluate document transcription and structured information extraction from real-world document images. It contains 1,000 English document images covering heterogeneous formats, including tables, charts, slides, forms, and identity documents. Each example is accompanied by document-format and image-quality metadata, along with structured JSON annotations and Markdown transcription targets. This design enables evaluation of both schema-constrained extraction and layout-aware document reconstruction.

A.2 PROMPTS AND METRICS

Prompt. We use each model's official chat template. System prompts follow each model's default, and the task instruction is identical across models. The complete Qwen2.5-VL template for each

task group is shown below; InternVL uses the same per-dataset task instruction, differing only in its default system prompt and image-placeholder format. Prompts are shown in a simplified, human-readable form, where <image> denotes the visual token placeholder.

Prompt Template: HTML-Target Datasets (Design2Code, WebCode2M, OCR-Md-Dense)

```
System: You are a helpful assistant.
User: <image>
Generate the HTML code for the webpage shown in the
screenshot. Output only the HTML.
Assistant:
```

Concretely, Design2Code, WebCode2M, and OCR-Md-Dense use the HTML instruction above, whereas OmniDocBench and OmniAI-OCR use the Markdown instruction below to preserve text, tables, and formulas.

Prompt Template: Document Datasets (OmniDocBench, OmniAI-OCR)

```
System: You are a helpful assistant.
User: <image>
Convert the document page in the image into Markdown,
preserving the text, tables, and formulas. Output only the
Markdown.
Assistant:
```

Metrics. We report three metrics following prior speculative-decoding work Li et al. (2024b); Hu et al. (2025). For one example, let G be the number of generated tokens after the prompt and let S be the number of target verification passes during decoding. **(1) Mean Accepted Tokens (MAT)** is the average number of generated tokens per verification pass,

$$\mathrm{MAT} = \frac{G}{S}, \tag{4}$$

where each verification pass contributes all tokens emitted in that iteration, including the root or bonus token, so plain autoregressive decoding gives $\mathrm{MAT} = 1$. We compute G/S for each example and report its mean over the examples in a benchmark. **(2) Throughput** is the decoding rate in generated tokens per second,

$$\mathrm{Throughput} = \frac{G}{T}, \tag{5}$$

where T is the measured online generation time. **(3) Speedup** is the ratio of a method's throughput to autoregressive (AR) throughput on the same filtered examples,

$$\mathrm{Speedup} = \frac{\mathrm{Throughput}}{\mathrm{Throughput}_{\mathrm{AR}}}. \tag{6}$$

We retain only examples that terminate with a natural EOS before the generation cap. The autoregressive baseline and the training-free retrieval methods are compared on their common natural-EOS subset; ViSpec is reported on its own natural-EOS subset because it uses a separate inference harness. For each benchmark and method, throughput and speedup are computed from the sums of G and T over that method's reported subset; the Average column in Table 1 is the arithmetic mean of the five benchmark values.

A.3 Reproduction Details of Compared Baselines

All experiments are conducted on a single NVIDIA RTX 4090 GPU using half-precision (`float16`) inference. We evaluate the methods with three representative VLMs covering different multimodal architectures: InternVL2.5 Chen et al. (2024)[1], Qwen2.5-VL Bai et al. (2025b)[2],

[1] https://github.com/OpenGVLab/InternVL

[2] https://huggingface.co/Qwen/Qwen2.5-VL-7B-Instruct

and Qwen3-VL Bai et al. (2025a)[3]. The training-free retrieval-based baselines (REST, PLD, SuffixD, and SAM-D) target text-only LLMs; we adapt each to the multimodal VLM setting. For a training-based reference we include ViSpec, a speculative-decoding method designed for VLMs; as it requires a target-specific trained draft model, we evaluate it only on Qwen2.5-VL, for which an official checkpoint is released. We describe each baseline and its implementation below.

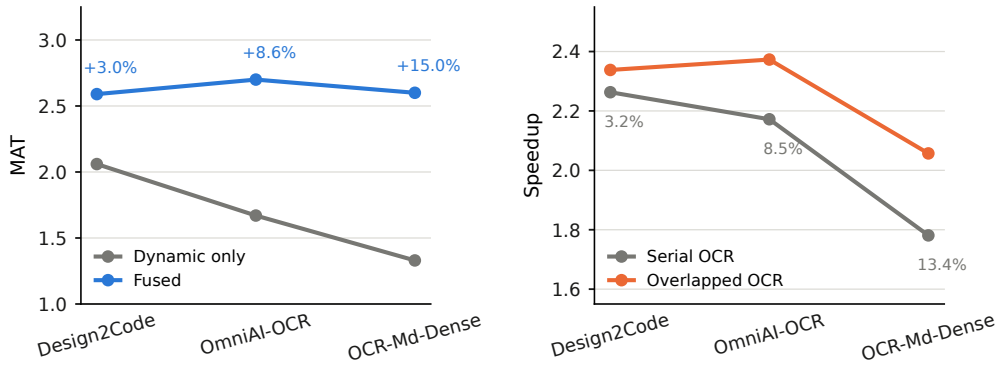
VTD configuration. VTD uses draft budget $N = 40$ nodes, expansion factor $\alpha = 4.0$, and per-depth branching cap $K = 8$ for the visible-text source (Appendix B.6 sweeps K). Both active sources use match-length offset $b_{\text{seq}} = b_{\text{vis}} = 0$; Appendix B.7 reports the effect of raising b_{vis} to the corpus-scale value. The OCR reader is EasyOCR with the English model; the recognized lines are joined with single spaces before tokenization. These settings are identical across all benchmarks and targets.

- **REST** (He et al., 2024)[4] **(NAACL 2024)**. REST performs retrieval-based speculative decoding without an auxiliary draft model. It constructs a suffix-array index over an external token datastore and retrieves entries whose prefixes match the longest suffix of the current decoding context. The retrieved continuations are organized into a frequency-pruned trie and verified by the target model using tree attention. We retrieve with the official multi-span schedule, trying suffix lengths $16, 15, \ldots, 2$ in descending order and taking the first span with a datastore hit, and prune the resulting trie to 64 candidate nodes by prefix frequency.
- **PLD** (Somasundaram et al., 2025)[5] **(NAACL 2025)**. Prompt Lookup Decoding (PLD) generates draft tokens by matching the most recent n-gram in the current context against earlier tokens in the prompt and previously generated prefix. When a match is found, PLD copies the corresponding continuation as a speculative draft, which is subsequently verified by the target model. This approach requires neither an external datastore nor an additional draft model. We use the paper's defaults, matching n-grams of size 3 down to 1 and copying up to 10 continuation tokens from the first match found.
- **SuffixDecoding** (SuffixD) (Oliaro et al., 2026)[6] **(NeurIPS 2025)**. SuffixDecoding is a model-free speculative decoding method that maintains suffix trees for both the active request and previously completed requests. At each decoding step, it matches the recent context against these trees and selects high-frequency continuations as draft tokens. The speculation length is adaptively determined by the matched suffix length, while low-confidence tokens are discarded before verification by the target model. Retrieval and tree construction use the official C++ implementation from ArcticInference with its default hyperparameters: maximum tree depth 64, speculation factor 1.0, speculation offset 0.0, minimum token probability 0.1, and tree-structured speculation enabled.
- **SAM-Decoding** (SAM-D) (Hu et al., 2025)[7] **(NeurIPS 2025)**. SAM-Decoding supports suffix-automaton retrieval over token sequences from the prompt, generated output, and optionally an external text corpus. During decoding, it identifies the longest suffix of the current context that appears in the automaton and uses the corresponding continuation as a draft sequence. Owing to the constant-time average suffix-matching operation, SAM-Decoding can efficiently retrieve long candidate sequences without an auxiliary draft model and the retrieved candidates are subsequently verified by the target model. We run it with a draft budget of $N = 40$ nodes, expansion factor $\alpha = 4.0$, per-depth branching cap $K = 8$, and match-length offset $b = 5$, emitting a draft tree rather than a linear sequence.
- **ViSpec** Kang et al. (2026)[8] **(NeurIPS 2025)**. ViSpec introduces a vision-aware speculative decoding framework with a lightweight multimodal draft module. It first applies a vision adaptor to compress the image representation while preserving the original positional information, and further extracts a global image feature that is injected into subsequent text representations. The resulting vision-conditioned drafter generates a tree of candidate tokens, which are verified by the target VLM in parallel. We follow the official inference

[3] https://github.com/qwenlm/qwen3-vl
[4] https://github.com/FasterDecoding/REST
[5] https://github.com/apoorvumang/prompt-lookup-decoding.git
[6] https://github.com/snowflakedb/ArcticInference
[7] https://github.com/hyx1999/sam-decoding
[8] https://github.com/KangJialiang/ViSpec

Table 4: Results on Design2Code, WebCode2M, OmniDocBench, OCR-Md-Dense, and OmniAI-OCR using Qwen3-VL-8B-Instruct as the target model. For each method we report the MAT, throughput (Tokens/s), and the speedup ratio over AR. Best results are in bold, suboptimal results are underlined.

Dataset	AR			SAM-D			VTD		
	MAT	Tokens/s	Speedup	MAT	Tokens/s	Speedup	MAT	Tokens/s	Speedup
Design2Code	1.00	44.57	1.00×	1.90	67.52	1.52×	2.31	79.59	1.79×
WebCode2M	1.00	44.61	1.00×	1.72	56.31	1.26×	2.53	77.97	1.75×
OmniDocBench	1.00	46.18	1.00×	1.39	53.34	1.16×	2.01	62.56	1.36×
OCR-Md-Dense	1.00	37.13	1.00×	1.36	31.86	0.86×	2.33	44.98	1.21×
OmniAI-OCR	1.00	46.13	1.00×	1.60	59.35	1.29×	2.54	84.44	1.83×



(a) Accepted length

(b) End to end speedup

Figure 4: Efficiency analysis on three representative datasets. (a) Mean accepted tokens (MAT) for dynamic drafting alone and for the fused drafter. Percentage annotations indicate the increase in verification time per step relative to dynamic drafting alone. (b) Speedup over autoregressive decoding with serial OCR accounting and under an idealized overlap assumption that fully hides OCR latency. OCR percentages indicate its share of serial latency. Connecting lines guide comparison across datasets and do not represent a continuous sweep.

setting with 2 query vectors, a draft-tree depth of 3, a branching factor of 8, and a total of 30 candidate tokens.

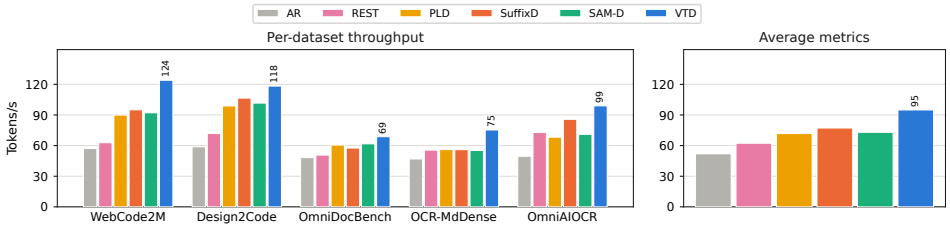
B Additional Experimental Results

B.1 Generalization to Qwen3-VL-8B

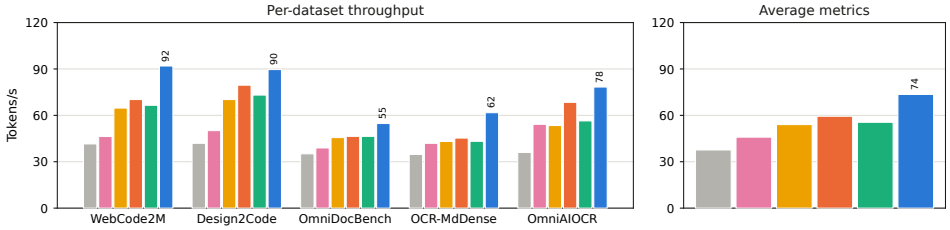
To assess generalization beyond the two target models evaluated in §5.2, we evaluate VTD on Qwen3-VL-8B-Instruct Bai et al. (2025a) across the same five benchmarks, with 100 fixed-seed evaluation examples per dataset, greedy decoding, 6144 token generation cap, and the natural-EOS filter. As shown in Table 4, VTD outperforms the in-sequence SAM-D baseline on all five benchmarks, improving end-to-end speedup by 17.2%–41.9% relative to this baseline. The largest gains occur on OCR-Md-Dense and OmniAI-OCR, consistent with the benefit of visible-text retrieval on text-rich document pages. These empirical results support the complementarity of the two retrieval sources on an additional target model.

B.2 Efficiency Analysis

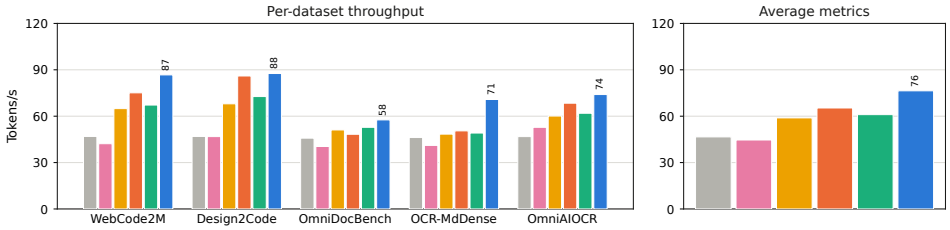
Figure 4 examines two costs that affect the conversion of accepted tokens into inference speedup: the additional verification work introduced by draft fusion and the OCR pass performed once per



(a) Throughput result on Qwen2.5-VL-3B-Instruct



(b) Throughput result on Qwen2.5-VL-7B-Instruct



(c) Throughput result on InternVL2.5-8B

Figure 5: Decoding throughput (generated tokens per second) across the Qwen2.5-VL-3B-Instruct (a), Qwen2.5-VL-7B-Instruct (b), and InternVL2.5-8B (c) targets, with each panel reporting VTDagainst the baselines on all five benchmarks. Higher is better. VTD attains the highest throughput on every dataset and target, and the margin widens as visible-text density grows.

request. We report these costs separately to distinguish improvements in drafting effectiveness from their effect on total latency.

Verification overhead. On Design2Code, fusion increases MAT from 1.82 to 2.37, while verification time per step increases by 3.0% relative to dynamic drafting alone. On OmniAI-OCR and OCR-Md-Dense, MAT increases from 1.67 to 2.70 and from 1.33 to 2.60, with corresponding increases in verification time per step of 8.6% and 15.0%. Across these datasets, the relative increase in accepted tokens exceeds the relative increase in verification cost. These results support the efficiency of verifying candidates from both retrieval sources in a single target model pass, while showing that the additional verification work has a measurable cost.

OCR latency and potential overlap. The OCR pass accounts for 3.2%, 8.5%, and 13.4% of serial latency on Design2Code, OmniAI-OCR, and OCR-Md-Dense, respectively. Under an idealized assumption that OCR latency is fully hidden without changing the remaining computation, the corresponding speedups would increase from 2.263× to 2.338×, from 2.172× to 2.373×, and from 1.781× to 2.057×. These estimates exclude OCR time from the serial total; they are not measurements of a concurrent serving implementation. Realizing this benefit would require sufficient independent work, such as visual prefill, and available compute resources to overlap OCR without slowing the target model. The visible text index must also be ready before it is used for drafting. Our main results retain serial OCR accounting.

Table 5: Settings with limited gains from visible text retrieval. Median tokens denotes autoregressive output length; speedup is relative to autoregressive decoding for VTD. Arabic-Md uses an English OCR model.

	Median	SAM-D		VTD	
Dataset	tokens	MAT	Speedup	MAT	Speedup
Low visible-text share					
ChartMimic	274	1.496	1.477×	1.566	1.508×
Chart2Code	310	1.903	1.848×	1.992	1.890×
Short outputs					
Web2Code (WebSight)	387	1.607	1.527×	1.763	1.621×
In-sequence drafter already strong					
Figma2Code	1015	2.431	2.340×	2.536	2.426×
PubTabNet	756	2.777	2.715×	2.955	2.807×
Negative control: wrong-language reader					
Arabic-Md (en. OCR)	530	1.322	1.283×	1.512	1.284×

B.3 Throughput Across Targets

Figure 5 reports throughput in generated tokens per second. VTD attains the highest throughput on every dataset and target, and no plotted baseline overtakes it. Reading throughput directly rather than as a ratio matters because the autoregressive baseline differs across targets: the 3B model decodes faster per token than the 8B models, so a fixed speedup ratio corresponds to a larger absolute gap on the smaller model. The panels make this difference concrete while preserving the per-target ranking.

The margin over the strongest baseline is substantial and consistent. Pooling the five benchmarks, VTD reaches $1.84\times$, $1.95\times$, and $1.64\times$ the autoregressive throughput on Qwen2.5-VL-3B, Qwen2.5-VL-7B, and InternVL2.5-8B, respectively, compared with $1.49\times$, $1.58\times$, and $1.40\times$ for SuffixD, the strongest baseline on each target. The advantage is not confined to the Qwen family: it also holds on InternVL2.5, whose multimodal architecture differs from that of Qwen2.5-VL. On this target, REST falls below the autoregressive baseline on three of the five benchmarks.

The relative advantage is particularly pronounced on OCR-Md-Dense, where VTD reaches $1.61\times$, $1.78\times$, and $1.53\times$ the autoregressive throughput on the three targets. The corresponding in-sequence baselines provide only limited acceleration, especially on the text-dense document pages. These results are consistent with the proposed mechanism: visible-text retrieval is most useful when a substantial fraction of the output is readable text from the input image. The advantage is generally smaller on webpage datasets, where the output also contains markup that is not visible in the image.

B.4 Applicability Boundary

Table 5 analyzes six settings where visible text retrieval yields marginal acceleration gains over dynamic SAM-D. Our results indicate that the utility of text retrieval hinges on three key factors: the presence of salient text within the image, the computational overhead of OCR relative to autoregressive generation, and the efficacy of historical sequence retrieval.

On ChartMimic and Chart2Code, VTD increases MAT by only 0.07 and 0.09, respectively. These small gains are consistent with outputs dominated by plotting code rather than transcribed text. Web2Code also shows a modest speedup increase, from $1.527\times$ to $1.621\times$; its short outputs (median 387 tokens) offer less decoding work over which to amortize OCR latency. On Figma2Code and PubTabNet, dynamic SAM-D already achieves MAT of 2.43 and 2.78, respectively, and adding visible text yields limited further gains. These patterns suggest several constraints on acceleration, although comparisons across datasets do not isolate their individual effects.

The Arabic-Md control illustrates the importance of OCR language coverage. With an English OCR model, MAT rises from 1.322 to 1.512, but speedup changes only from $1.283\times$ to $1.284\times$. Thus, increased acceptance alone need not offset OCR latency. Together, these results indicate that applicability depends on recoverable text and total inference cost, rather than task category alone.

Table 6: WebCode2M results stratified by output length on Qwen2.5-VL-7B. The test set is partitioned into its **Short**, **Middle**, and **Long** subsets (median autoregressive output 1,162 / 1,771 / 2,203 tokens). For each method we report the MAT and speedup ratio over AR. † denotes training-based methods. Best results are in bold, suboptimal results are underlined.

Model	Method	Short		Middle		Long	
		MAT	Speedup	MAT	Speedup	MAT	Speedup
Qwen2.5-VL-3B	AR	1.00	1.00×	1.00	1.00×	1.00	1.00×
	REST	1.18	1.07×	1.18	1.07×	1.26	1.15×
	PLD	1.66	1.51×	1.69	1.52×	1.90	1.67×
	SuffixD	2.14	1.65×	2.11	1.54×	2.68	1.80×
	SAM-D	1.76	1.53×	1.90	1.55×	2.31	1.74×
	VTD	2.72	2.16×	2.92	2.18×	3.08	2.16×
Qwen2.5-VL-7B	AR	1.00	1.00×	1.00	1.00×	1.00	1.00×
	ViSpec†	2.69	1.41×	2.52	1.31×	2.85	1.53×
	REST	1.21	1.10×	1.19	1.10×	1.24	1.15×
	PLD	1.64	1.55×	1.63	1.55×	1.68	1.59×
	SuffixD	2.00	1.71×	1.86	1.60×	2.18	1.81×
	SAM-D	1.66	1.55×	1.71	1.59×	1.81	1.68×
	VTD	2.51	2.23×	2.54	2.23×	2.58	2.24×
InternVL2.5-8B	AR	1.00	1.00×	1.00	1.00×	1.00	1.00×
	REST	1.11	0.89×	1.13	0.89×	1.16	0.92×
	PLD	1.75	1.41×	1.75	1.39×	1.69	1.35×
	SuffixD	2.19	1.62×	2.11	1.57×	2.19	1.62×
	SAM-D	1.84	1.46×	1.80	1.42×	1.77	1.42×
	VTD	2.45	1.87×	2.52	1.89×	2.38	1.79×

B.5 Robustness to Output Length

Table 6 stratifies WebCode2M by autoregressive output length. On Qwen2.5-VL-7B, VTD maintains nearly constant end-to-end speedup across the short, middle, and long subsets ($2.23\times$, $2.23\times$, and $2.24\times$), while its MAT changes only from 2.51 to 2.58. This stability is consistent with the visible-text index being constructed before decoding and therefore being available from the first generated token.

The history-based baselines show a stronger dependence on the generated prefix. For example, SuffixD improves from $1.71\times$ on the short subset to $1.81\times$ on the long subset, while PLD improves from $1.55\times$ to $1.59\times$. Consequently, on the Qwen targets, VTD's relative advantage is largest on the short subset, where history-based methods have less context to exploit. This pattern is model-dependent, however: on InternVL2.5-8B, the largest relative advantage over SAM-D occurs on the middle subset. Thus, the results support stable performance across output lengths rather than a universal monotonic relationship between output length and relative gain.

B.6 Tree Width Sensitivity

The per-depth branching cap K controls the coverage of alternative continuations in the merged draft tree and therefore its verification width. Table 7 sweeps $K \in \{4, 8, 16, 32\}$ on Qwen2.5-VL-7B for Design2Code and OCR-Md-Dense. On Design2Code, increasing K from 4 to 8 improves MAT from 2.591 to 2.639 and speedup from $2.263\times$ to $2.295\times$; increasing it further gives no additional benefit. On OCR-Md-Dense, even $K = 4$ reaches the same plateau, with speedup varying by only $0.001\times$ across the sweep. Thus, $K = 8$ provides the useful branch coverage while avoiding the extra verification width of larger trees. We use this operating point throughout.

Table 7: Tree-width sweep on Qwen2.5-VL-7B. $K = 8$ reaches the useful branch-coverage plateau; larger values add negligible benefit.

	Design2Code		OCR-Md-Dense	
K	MAT	Speedup	MAT	Speedup
4	2.591	2.263×	2.598	1.786×
8	2.639	2.295×	2.600	1.786×
16	2.593	2.264×	2.600	1.786×
32	2.593	2.264×	2.600	1.787×

Table 8: Match-length offset b_{vis} on Qwen2.5-VL-7B. The corpus-scale offset b_{vis}=5 suppresses short visible-text matches and costs MAT on every dataset; VTD uses b_{vis}=0 because the index holds a single page, where short matches are evidence rather than noise.

Params	WebCode2M			Design2Code			OmniDocBench			OCR-Md-Dense			OmniAI-OCR		
	MAT	Speedup	Δ	MAT	Speedup	Δ	MAT	Speedup	Δ	MAT	Speedup	Δ	MAT	Speedup	Δ
b_{vis}=5	2.33	1.89×	-7.8%	2.01	1.86×	-15.2%	1.58	1.41×	-22.4%	1.71	1.36×	-34.2%	1.87	1.65×	-30.7%
b_{vis}=0 (VTD)	**2.52**	**2.22**×	–	**2.37**	**2.13**×	–	**2.03**	**1.52**×	–	**2.60**	**1.78**×	–	**2.70**	**2.17**×	–

B.7 Match-Length Offset

The offset b_{vis} in Eq. 1 discounts short suffix matches before allocating draft nodes, suppressing candidates from matches shorter than the offset. Table 8 compares the value used by VTD ($b_{\text{vis}} = 0$) with the corpus-scale value that SAM-D applies to its external static index ($b_{\text{static}} = 5$). Transferring the larger offset to the visible-text source reduces MAT on every benchmark, by 7.8% on WebCode2M and 34.2% on OCR-Md-Dense, with a corresponding loss of speedup.

The asymmetry follows from index size. A static corpus spans many documents, so a short match is weak evidence about the current request and the offset filters spurious candidates. The visible-text index holds the text of a single page, where a two- or three-token match already localizes a specific position in that page. Discounting such matches removes most of the source's candidates and returns the method toward its dynamic-only behavior. This distinction is a property of the index rather than of the retrieval mechanism, which is why VTD sets the two offsets independently.

B.8 OCR Language Coverage

Fusion both enlarges the verified draft tree and supplies image-specific text, so a speedup gain alone does not identify which factor is responsible. Mismatching the reader's language to the page separates them: the tree is constructed exactly as before, but its visible-text candidates no longer correspond to the image.

Table 9 reports three conditions on Qwen2.5-VL-7B. An English reader on the English subset of OmniDocBench and on the full bilingual benchmark gives speedup gains of 16.6% and 16.1% over dynamic-only retrieval. An English reader on Arabic pages (Arabic-Md) leaves speedup unchanged at $1.284\times$ while still raising MAT from 1.322 to 1.512: the recognized text is unrelated to the page, so its candidates occupy tree capacity and are then rejected. Additional candidates therefore do not accelerate decoding by themselves, and the gains reported in §5.2 require OCR text that matches the target's output.

B.9 Case Studies

Per-page source attribution. The aggregate results show that visible-text retrieval drives much of the gain. To make this mechanism concrete, we instrument the Qwen2.5-VL-7B decoder and attribute each accepted token to one of three categories: the verifier's one-token bonus, the in-sequence dynamic index, or the per-image visible-text index. The following examples illustrate how these sources compose on individual pages. Figure 6 shows an OCR-Md-Dense page over 122

Table 9: Content-match control on Qwen2.5-VL-7B. An English reader on Arabic pages returns text that does not match the image, and the gain vanishes, isolating OCR *content* rather than a wider tree. Δ vs SAM-D is the speedup gain over dynamic-only retrieval. Rows 1–2 use the English-only and full bilingual OmniDocBench subsets, so neither matches its column in Table 1.

Condition	Dataset	OCR lang	VTD speedup	Δ vs SAM-D
matched	OmniDocBench (en)	en on en	1.67×	+16.6%
matched	OmniDocBench	en on bilingual	1.52×	+16.1%
wrong-lang	Arabic-Md	**en on Arabic**	1.25×	-1.6%

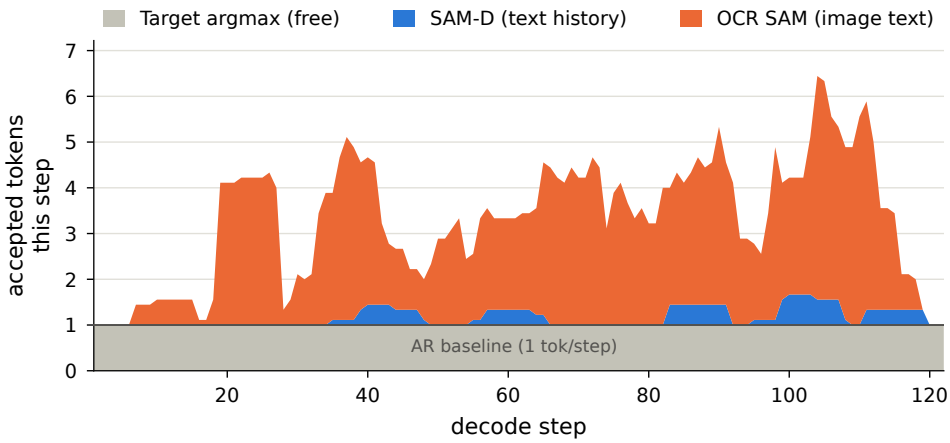


Figure 6: Per-step accepted length on one OCR-Md-Dense page, decomposed by source. The gray band is the verifier's one-token bonus, the blue band is the in-sequence dynamic index, and the orange band is the visible-text index built from OCR.

verification steps. The fused run has MAT 3.33, decomposed into 1.00 verifier-bonus tokens, 0.16 tokens from the dynamic index, and 2.16 tokens from the visible-text index per verification step. The visible-text source contributes across most of the page, although its per-step contribution varies with the generated context. In concrete terms, the visible-text index nearly triples the in-sequence drafter's accepted length on this page (1.17 $\rightarrow$ 3.33 tokens per step), raising the single-page verify speedup from 1.12× to 2.39× over autoregressive decoding.

Where the accepted tokens come from. Figure 7(a) renders the opening of the same page's output, the first 185 generated tokens, with each token tinted by the source that proposed it. The page shows a clean division of labour. The *structural* tokens are the target model's own argmax (gray): the code fence ` ```html `, the `<html><body>` and `<p>` tags, and the markup that closes each paragraph. None of these appear as text in the image, so the visible-text index cannot supply them; the model generates them itself, one per step, as the free verifier bonus. The *content* tokens are supplied by the visible-text index (orange) in long verbatim runs read straight off the page, and the two sources interleave at a fine grain: where the tokenizer splits a word across the markup/text boundary (`S, USING`; `&E AND`), the target supplies the fragment and the index resumes, so a single decode step routinely mixes markup the model wrote with a sentence it read. The drafter is not guessing structure; it is replaying text physically present in the input, which is why the accepted runs are long and correct, while the markup that holds the document together is still produced by the target model itself.

Panels (b) and (c) show that the same mechanism holds at higher text density and on a second dataset. On the statistical results section of panel (b) (page-mean MAT 7.05, of which 4.92 accepted tokens per step come from the visible-text index), the index reads off the findings almost continuously, `estream samples with strain TA98. However, with, 86E MS prototype was statistically higher than the, the Marlboro Light SS and the Camel Light`, while the target contributes the fragments the tokenizer splits (`among`, `sid`, `Camel`) and the dynamic index replays an entire section header the page established earlier

```
```html
<html><body>
<p>POSTER 1011 : ABSTRACT</p>
<p>HISTOPATHOLOGICAL EXAMINATIONS WERE CONDUCTED ON ALL RESPIRATORY TRACT ORGANS
OF ALL  ANIMALS, USING H& E AND PERIODIC ACID SCHIFF ALCIAN BLUE STAINS. ALL
OTHER MAJOR TISSUES WERE EXAMINED FOR HIGH DOSE AND CONTROL GROUPS. ALL SLIDES
WERE READ WITHOUT KNOWLEDGE OF TREATMENT GROUP.</P>
<p>RESPONSE OBTAINED IN THE REFERENCE (TOBACCO BURNED) GROUPS WERE SIMILAR TO
PUBLISHED OBSERVATIONS. IN THE TEST CIGARETTE (TOBACCO NOT BURNED) GROUPS THE
FOLLOWING RESPONSES WERE EITHER ABSENT.
```

(a) OCR-Md-Dense, the top of a poster abstract. The target model writes the markup scaffold (“‘html, <html>/<body>, the <p> tags); the visible-text index fills each paragraph with the sentences read off the page.

```
from a market Marlboro Light 85; and (c) Mainstream and Sidestream particulate
from a market Camel Light 85.</p>
<p>SIGNIFICANCE: No statistical differences were detected among the mainstream
or among the sidestream samples with strain TA98. However, with TA100 the
CT2286E MS prototype was statistically higher than the Camel Light MS; and, the
CT2286E SS prototype was statistically higher than both the Marlboro Light SS
and the Camel Light SS.</p>
<p>
```

(b) OCR-Md-Dense, a statistical results section. Long orange runs transcribe the findings verbatim, while the in-sequence dynamic drafter (blue) replays a whole section header (`SIGNIFICANCE: No statistical differences were detected`) that the page has already established.

```
[street address], State of California.

2. **TERM OF LEASE**

The term of this Lease shall be for period of 3 year(s) 7 month(s) commencing on
the 8 days of September 2025 day of September 2025, and expiring at Midnight on
the 8 day of April 2029 day of April 2029. (“Initial Term”)

3. **BASE RENT:**

The net monthly payment shall be 3655 dollars ($3655) per month.
```

(c) OmniAI-OCR, a lease contract from a different benchmark, spanning two numbered clauses. All three sources interleave—the index transcribes the fixed contract phrasing, the dynamic drafter replays the recurring boilerplate (`**`, `The`, dates), and the target supplies structure and split digits.

Figure 7: Token-level source attribution on three pages, each token coloured by the source that proposed it: target argmax (gray), in-sequence dynamic index (blue), and per-image visible-text index (orange). Across text density (a, b) and across datasets (a, b vs. c), the same division of labour holds: the target model produces the markup structure that is not visible in the image, the visible-text index supplies the on-page content in long verbatim runs, and the dynamic index replays whatever the page repeats. The sources compose inside single decode steps.

(`SIGNIFICANCE: No statistical differences were detected`). This is the same behaviour as panel (a), with a larger visible-text share and hence a longer accepted run: the in-sequence drafter alone reaches only 1.68 tokens per step here, so the visible-text index lifts the accepted length more than fourfold ($1.68 \rightarrow 7.05$) and the single-page verify speedup from $1.56\times$
```

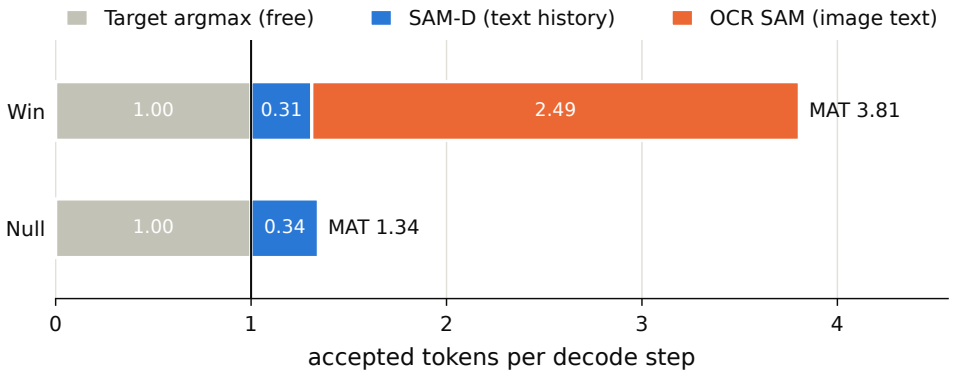


Figure 8: Two WebCode2M pages, whole-page mean accepted tokens per step by source. On the win case the visible-text index (orange) contributes the majority of the accepted length; on the null case it contributes nothing and VTD coincides with the in-sequence drafter. Same page, same method—the difference is how much of the output is visible text.

to $4.74\times$. Panel (c) is a lease contract from the OmniAI-OCR benchmark (`omni_227`, MAT 3.70), whose page style and document structure differ from the Markdown reports. Here all three sources interleave across two numbered clauses: the index transcribes the fixed contract phrasing (`shall be for a period of 3 year(s), iring at Midnight on the 8, monthly payment shall be`), the target fills the connecting words and split digits (`month`, `encing`, `day of`), and the dynamic drafter (blue) supplies the short tokens the contract's own boilerplate repeats (`**`, `The`, `of April`, `2029`). The gain is not confined to one document type or one dataset: wherever the output is text the model can see, the visible-text index supplies it verbatim, and the target is left to produce only the structure and the few fragments the transcription cannot.

When the visible-text index does nothing. The gain is contingent, not uniform. Figure 8 contrasts two WebCode2M pages. On the first, the visible-text index supplies 2.50 of the 3.81 accepted tokens per step. On the second page , the index lands *nothing*: the page's output is dominated by markup and styling with little on-screen text, so accepted length is 1.34—identical to the in-sequence drafter alone, since with no visible-text contribution VTD reduces to it. This is the per-page form of the applicability boundary of §B.4: VTD accelerates a page exactly to the extent that the page's output is text the model can see.

C Algorithmic Details

Algorithm 1 describes greedy decoding with the dynamic and visible text sources. We distinguish the emitted sequence y from a pending token r selected by the target model. This token becomes the shared root of the next verification tree and is emitted only when that tree is processed.

Algorithm 1 VTD: greedy decoding with two retrieval sources

Require: Image I, prompt tokens x, target p, OCR model Φ, token limit L, KV-cache capacity $C_{\max}$
Require: Draft parameters N, α, K and source offsets b_s
Ensure: Generated token sequence y

$y \leftarrow [\,]$
$\mathcal{A}_{\text{seq}} \leftarrow \textsc{BuildDynamicSAM}(x)$
$\mathcal{A}_{\text{vis}} \leftarrow \textsc{BuildSAM}(\textsc{Tokenize}_p(\Phi(I)))$
Reset the visible match state and advance it through x
$(C, r) \leftarrow \textsc{Prefill}(p, I, x)$
while $|y| < L$ and y contains no stop token **do**
  Stop if $\text{len}(C) + (2N - 1) > C_{\max}$
  for $s \in \{\text{seq}, \text{vis}\}$ **do**
    $(i_s, \ell_s) \leftarrow \mathcal{A}_s.\textsc{Lookup}(r)$
    Compute n_s using Eq. 1
    $\mathcal{T}_s \leftarrow \textsc{GrowTree}(\mathcal{A}_s, i_s, r, n_s, K)$
  end for
  $\mathcal{T} \leftarrow \textsc{MergeRoots}(\mathcal{T}_{\text{seq}}, \mathcal{T}_{\text{vis}})$
  $(z, r_{\text{next}}, J, C^+) \leftarrow \textsc{VerifyGreedy}(p, C, \mathcal{T})$
  $z \leftarrow \textsc{Truncate}(z, L - |y|, \text{stop tokens})$
  $C \leftarrow \textsc{KeepPath}(C^+, J_{1:|z|})$
  $y \leftarrow y \circ z$
  $\mathcal{A}_{\text{seq}}.\textsc{Extend}(z)$
  $\mathcal{A}_{\text{vis}}.\textsc{Advance}(z)$
  $r \leftarrow r_{\text{next}}$
end while
return y

State and verification. The retrieval prefix x excludes image placeholders; PREFILL processes the full multimodal input, initializes the key and value cache C, and selects the first root r. LOOKUP advances the match state tentatively with r without modifying it. Tree construction and fusion follow §4.2. In one target pass, VERIFYGREEDY returns the longest accepted path z including r, its node indices J, the temporary cache C^+, and the next greedy token r_{next}; path ties use a fixed ordering. KEEPPATH retains the original context and accepted nodes in sequence order. Only emitted tokens extend the dynamic index and advance the visible match state. Let $C_{\max}$ denote the allocated KV-cache capacity in tokens; the algorithm stops before drafting when the current cache length plus the maximum merged tree size $2N - 1$ would exceed this capacity. MAT includes the current root.

Stopping and correctness. TRUNCATE retains tokens through the first EOS or configured stop token, subject to the remaining token budget. Truncation ends decoding, so any unused next root is discarded. The root guarantees progress even if all descendants are rejected. Under identical conditional logits and deterministic argmax tie breaking, induction over accepted tokens establishes equivalence to greedy autoregressive decoding. This argument does not guarantee bitwise identity under finite precision.