

HSD: Training-Free Acceleration for Document Parsing Vision-Language Models with Hierarchical Speculative Decoding

Wenhui Liao^{1,2†}, Hongliang Li^{1,2†}, Pengyu Xie^{2,4}, Xinyu Cai²,
Yufan Shen², Yi Xin², Qi Qin², Shenglong Ye², Tianbin Li²,
Ming Hu², Junjun He², Yihao Liu², Wenhai Wang², Min Dou²,
Bin Fu^{2,3*}, Botian Shi^{2*}, Yu Qiao^{2*}, Lianwen Jin^{1*}

¹ South China University of Technology, Guangzhou, China

² Shanghai Artificial Intelligence Laboratory, Shanghai, China

³ Shenzhen Institute of Advanced Technology, CAS, Shenzhen, China

⁴ Nanjing University, Nanjing, China

{eelwh,eehongliangli}@mail.scut.edu.cn

{fubin,shibotian,qiaoyu}@pjlab.org.cn

eelwj@scut.edu.cn

Abstract. Document parsing is a fundamental task in multimodal understanding, supporting a wide range of downstream applications such as information extraction and intelligent document analysis. Benefiting from strong semantic modeling and robust generalization, VLM-based end-to-end approaches have emerged as the mainstream paradigm in recent years. However, these models often suffer from substantial inference latency, as they must autoregressively generate long, full-page sequences when processing long-form documents. While recent hybrid methods mitigate this issue via region-level parallel decoding with VLMs, independent region decoding loses full-page context and might weaken global coherence. To address this issue, we propose **Hierarchical Speculative Decoding (HSD)**, a two-stage local-to-global framework for document parsing. HSD first employs a lightweight pipeline drafter to predict region partitions and generate coarse drafts for each region. The first stage verifies the generated region-level drafts in parallel for efficiency, while the second stage further performs page-level verification on these refined outputs to preserve full-page coherence. Experimental results show that our HSD achieves a near-lossless $2.78\times$ speedup with HunyuanOCR on OmniDocBench v1.5 and up to $7.04\times$ speedup on long-document parsing tasks, demonstrating the effectiveness of our proposed method. The code is available at <https://github.com/whlscut/HSD>.

Keywords: Document Parsing · Hierarchical Speculative Decoding · Vision–Language Models

[†] Equal contribution.

^{*} Co-corresponding authors.

1 Introduction

Document parsing [4, 39, 49] converts document images into structured text by organizing text, formulas, tables, and figures in their natural reading order. As a foundational technology, it underpins a broad spectrum of downstream applications, including document indexing and retrieval, workflow automation, data governance, and large-scale corpus construction. To enable these applications at scale, document parsing systems are expected to process massive volumes of documents with high accuracy and efficiency. However, achieving both objectives simultaneously remains challenging in practice.

Recent progress in document parsing can be divided into three categories: pipeline-based [27, 32, 41], end-to-end [4, 33, 43], and hybrid [13, 23, 30] approaches. Pipeline-based methods decompose the workflow into sub-tasks (e.g., layout analysis [48], reading-order prediction, text recognition, formula recognition, and table parsing), each handled by a lightweight specialist. This design enables region-level parallel recognition after layout segmentation, thereby improving efficiency. However, the overall performance is often limited by lightweight recognizers, and errors can propagate across stages. End-to-end approaches are typically built on vision-language models (VLMs) [1, 8, 20, 25], which directly generate the full parsing results for the input image. Benefiting from VLMs’ strong semantic modeling and the global coherence maintained during generation, these methods are usually more robust to complex layouts, noise, and cross-region dependencies. However, autoregressive decoding makes long outputs inherently slow, causing latency to grow roughly linearly with sequence length. Hybrid approaches aim to combine both paradigms: they first perform layout analysis to segment a page into semantic regions, and then decode these regions independently in parallel with the VLM. While region parallelism improves efficiency, independence across each individual region removes cross-region correlation (e.g., reading order, cross-column connections), which might damage page-level coherence. Moreover, once the predicted layout or reading-order priors are wrong, the VLM is forced to decode under incorrect partitions/orders, resulting in error propagation. In summary, existing approaches still struggle to simultaneously exploit region-level parallelism for efficiency while preserving page-level global coherence for robustness.

In this work, we show that the conflict between region-level parallelism and page-level global coherence is not inherent. To this end, we introduce Hierarchical Speculative Decoding (HSD), a new paradigm for end-to-end document parsing that performs two-stage speculative verification: coarse drafts are first verified in parallel at the region level, and then the refined outputs are verified globally at the page level to restore full-page coherence. This design is motivated by the structured layout of documents, where content can be segmented into semantic regions such as paragraphs, tables, and figures. Concretely, a lightweight pipeline model first produces a semantic region partition together with coarse predictions for each region, which serve as speculative drafts. In the region-level verification stage (Stage 1), these drafts are verified in parallel by the end-to-end parser. However, since this stage lacks full-page context and may inherit

layout segmentation errors from the pipeline parser, it will introduce structural inconsistencies such as incorrect layout hierarchy or reading order. To address this problem, the page-level verification stage (Stage 2) performs full-page verification conditioned on the outputs of Stage 1. Since these outputs have already been refined, page-level verification requires only a moderate number of steps to modify the remaining structural errors.

To further speed up the verification, we propose Decoupled Speculative Verification (DSV) for HSD. Unlike traditional speculative decoding, which repeatedly refreshes draft tokens to maintain prefix synchronization, we directly reuse the pipeline’s region predictions generated in a single forward pass as drafts. While it greatly reduces draft generation cost, such decoupling introduces prefix-draft misalignment between the pre-generated drafts and the VLM’s current generation prefix. DSV utilizes a *draft-target matching process* to address this alignment problem, and the *prefix-tree batching mechanism* further enables efficient verification over multiple candidate matched segments.

To demonstrate the effectiveness of our method, we evaluate HSD on OmniDocBench v1.5 [30], olmOCR-Bench [33], and Ocean-OCR-Bench [7] using multiple end-to-end parsers, including specialized document VLM parsers (dots.ocr [36], HunyuanOCR [38]) and general-purpose VLMs (Qwen2.5-VL-3B/7B [3] and Qwen3-VL-2B/8B [2]). Experimental results demonstrate that our method delivers near-lossless acceleration across models, document types, and languages. In particular, with HunyuanOCR, our method achieves end-to-end speedups of $2.78\times$, $2.46\times$, and $3.29\times$ across OmniDocBench v1.5, olmOCR-Bench, and Ocean-OCR-Bench, respectively.

Our main contributions are threefold:

- We introduce **Hierarchical Speculative Decoding** for end-to-end document parsing, a paradigm that exploits region-wise parallel verification, and restores global coherence via page-level verification.
- We propose **Decoupled Speculative Verification** for further acceleration, introducing a draft-target matching process to resolve misalignment and the prefix-tree batching mechanism to efficiently verify multiple draft segments.
- Experiments across parsers and benchmarks show that our **training-free, plug-and-play** method achieves up to $7.04\times$ speedup with near-lossless accuracy, highlighting a promising approach for efficient document parsing.

2 Related Work

2.1 Document Parsing

Traditional Pipelines. Document parsing [28, 32, 41] is inherently heterogeneous: pages mix text, mathematical expressions, tables, charts, and figures arranged in widely varying layouts and reading orders. To address these challenges, early systems decomposed the task into submodules—layout analysis, reading-order estimation, text recognition, formula recognition, table recognition, and

final assembly. Representative toolkits such as Marker [32], MinerU [41], PP-StructureV3 [11], and Docling [27] exemplify this design. For example, MinerU [41] begins with layout detection to partition a page into semantically labeled regions. It then routes each region to task-specific recognizers (text, equations, tables) and reconstructs the reading order to produce a consolidated Markdown result. These pipelines deliver practical throughput and engineering flexibility: lightweight specialists process layout regions in parallel, and modules can be swapped without retraining. However, cross-stage error propagation and limited submodule robustness can lead to failures on complex documents.

End-to-End Approaches. Recent work [9, 26] has shifted toward end-to-end document parsing with a single autoregressive vision–language model (VLM). The model ingests a page image and directly emits structured markup, jointly modeling text, tables, equations, and reading order under a unified objective and long-context decoding. Early efforts such as Nougat [4] targeted scientific articles, converting pages into lightweight LaTeX/Markdown-style markup. GOT [43] generalizes this paradigm to broader document types and richer elements (e.g., molecular formulas, sheet music, geometric shapes, and charts) while emphasizing efficiency via a highly compressed vision encoder paired with a 0.5B decoder. Subsequent models such as Ocean-OCR [7], olmOCR [33], dots.ocr [36], and HunyuanOCR [38] push accuracy by adopting native-resolution encoders and training on larger, higher-quality, and more diverse corpora. Beyond supervised learning, researchers have also begun to explore reinforcement learning and verifiable training signals: Infinity-Parser [40] optimizes layout fidelity with rewards on edit distance, paragraph counts, and structural consistency, whereas olmOCR 2 [34] uses binary unit tests as a programmatic reward signal. More recently, DeepSeek-OCR [44] reduces token budgets by compressing visual context into fewer vision tokens before text decoding, shortening sequences and latency. Overall, end-to-end parsers now achieve competitive parsing accuracy, yet autoregressive decoding over long outputs remains a key inference bottleneck.

Hybrid Approaches. Hybrid parsers [10, 13, 23, 30] combine the efficiency of pipelines with the semantic capacity of end-to-end models. They first run a lightweight layout stage to segment and order semantic regions on the page, then parse each region with a vision–language model, and finally stitch the results into full-page markup. Representative systems include MonkeyOCR [23], Dolphin [13], MinerU2.5 [30], and PaddleOCR-VL [10]. Such designs reduce the effective sequence length per decoding call and exploit concurrency, but token generation within each region remains autoregressive, so long regions can still be latency-dominant. Moreover, these methods can be sensitive to the quality of the layout analysis stage: boundary, ordering, or granularity mismatches between region decomposition and generation may propagate to the final parsing, potentially affecting cross-region coherence (e.g., multi-column flow and table continuity). Finally, the multi-stage pipeline introduces additional interfaces and objectives, making holistic end-to-end optimization more involved than in single-model parsers.

2.2 Speculative Decoding

Speculative decoding [6, 19, 37, 50] accelerates autoregressive generation with a draft–verify scheme: a fast drafter proposes multiple next tokens, and the target model verifies them in a single forward pass using rejection sampling, preserving the target distribution while reducing sequential steps. Existing methods can be broadly classified along two axes. The first axis is who drafts: external drafters use a separate smaller model; internal drafters augment the target model with lightweight drafting modules, including Medusa [5], EAGLE [22], and EAGLE-2 [21]; self-speculative methods [12], also called early-exit, allow the same network to draft with truncated layers and then verify with the full stack. The second axis concerns what is drafted: linear chunks or a branching tree [29]. In linear drafting, the drafter predicts a single k -token continuation, and the verifier accepts the longest matching prefix before the first mismatch. In tree drafting [29], the drafter explores multiple alternatives at one or more steps, forming a token tree that is verified in parallel; this typically increases per-round acceptance, especially for long outputs, at the cost of additional draft compute and memory. Extensions to VLMs [14, 18, 24, 46, 47] adapt these ideas to visual inputs and report around twofold throughput gains on systems such as LLaVA while maintaining quality. Despite this progress, speculative decoding remains underexplored for document parsing, where outputs are typically long and highly structured.

3 Methodology

Hierarchical Speculative Decoding (HSD) is a training-free, inference-time acceleration method for end-to-end document parsers. As shown in Fig. 1, a lightweight document parsing pipeline first performs layout analysis and element recognition (e.g., text/table/formula) to construct coarse drafts for document regions. The end-to-end parser then verifies these drafts hierarchically in two stages: (1) a region-level local verification stage that verifies drafts on cropped regions in parallel, producing refined region-level parses that are aggregated into a page draft; (2) a page-level global verification stage that verifies the page draft with modest multi-token decoding steps to obtain the final page-level parsing. In the following, we first introduce this two-stage hierarchical paradigm in Section 3.1, and then provide details of our verification operator in Section 3.2.

3.1 Hierarchical Speculative Decoding Paradigm

Setup and Notation. Given the page image x , the end-to-end parser p_θ autoregressively produces tokens y_t forming a sequence $\mathbf{y} = (y_{1:T})$ that spans text, formulas, tables, and figure markers, with conditional probabilities $p_\theta(y_t|x, y_{<t})$. A lightweight pipeline q_ϕ runs once per page and outputs a page layout $\mathcal{R} = \{r_i\}_{i=1}^M$ with a set of fixed drafts $\tilde{\mathcal{Y}}^{(i)}$ for each region r_i . We denote the verification operator by SpecDecode:

$$\hat{\mathbf{y}} = \text{SpecDecode}(p_\theta, z, \tilde{\mathcal{Y}}), \quad (1)$$

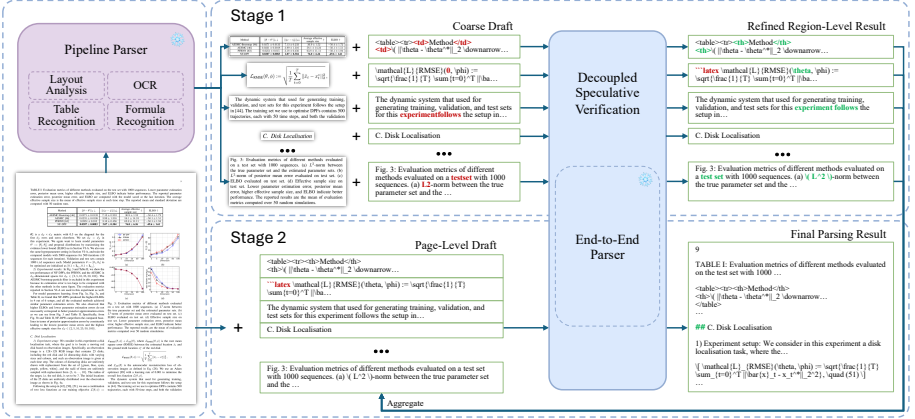


Fig. 1: Overview of the proposed hierarchical speculative decoding paradigm. A lightweight pipeline generates coarse region drafts via layout analysis and element recognition. The end-to-end parser verifies these drafts in two stages: Stage 1 performs region-level verification on cropped regions in parallel to produce refined drafts; Stage 2 aggregates them into a page-level draft and performs global, page-level verification to produce the final parsing result. Red and green text denote draft errors and their corrections during verification, respectively.

which takes the current visual input z (either a region crop or the full page) along with a draft set $\tilde{\mathcal{Y}}$, and returns a verified sequence $\hat{\mathbf{y}}$. SpecDecode treats drafts as proposals and uses the document parser p_θ to accept or correct them in several forward passes (details in Section 3.2). Building on this operator, we instantiate a two-stage hierarchy in which the first stage verifies region drafts in parallel, and the second stage performs a single page-level pass to reconcile context.

Stage 1 (Region-level Local Verification). For each $r_i \in \mathcal{R}$, let $z_i = x|_{r_i}$ denote the cropped region. We verify the corresponding region drafts in parallel:

$$\hat{\mathbf{y}}^{(i)} = \text{SpecDecode}(p_\theta, z_i, \tilde{\mathcal{Y}}^{(i)}). \quad (2)$$

Parallel region-wise verification provides high throughput. However, since this stage lacks full-page context and may inherit layout segmentation errors from the pipeline parser, it can introduce structural inconsistencies, such as incorrect layout hierarchy or reading order.

Stage 2 (Page-level Global Verification). To address these residual errors, we aggregate Stage 1 outputs into an unordered collection as a page-level draft,

$$\tilde{\mathcal{Y}}^{\text{pg}} = \left\{ \hat{\mathbf{y}}^{(i)} \mid r_i \in \mathcal{R} \right\}, \quad (3)$$

which is also kept fixed during verification.

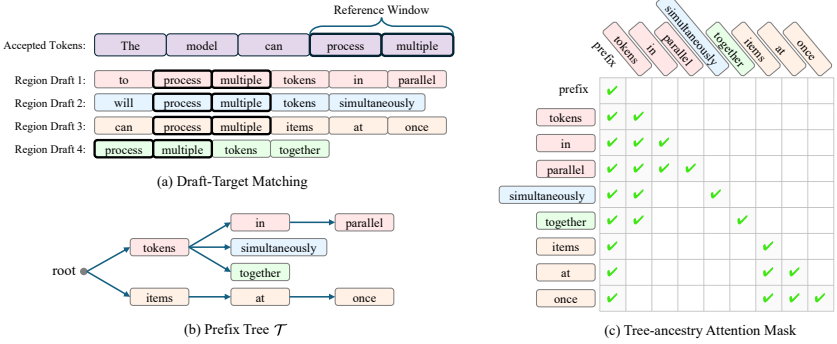


Fig. 2: Visualization of decoupled speculative verification. (a) Draft–target matching aligns a reference window from the accepted tokens with multiple drafts to extract candidate continuations. (b) The prefix tree organizes candidates by merging common prefixes. (c) The tree-ancestry attention mask enables parallel verification, where tokens attend only to the accepted tokens and their ancestors in the prefix tree (green checkmarks indicate allowed attention).

Then, we perform a full-page verification on the document:

$$\hat{\mathbf{y}}^{\text{pg}} = \text{SpecDecode}(p_\theta, x, \tilde{\mathbf{y}}^{\text{pg}}). \quad (4)$$

The final reading order is resolved by p_θ during verification. In other words, Stage 1 outputs serve as high-quality page-level drafts, allowing Stage 2 to complete verification in fewer forward steps.

3.2 Decoupled Speculative Verification

Traditional speculative decoding refreshes drafts synchronously at each decoding step, so the draft tokens are conditioned on the currently accepted tokens of the target model. In contrast, our setting is *decoupled*: a lightweight pipeline generates the coarse drafts *once per page*, which remain unchanged during verification. This decoupling introduces misalignments between the pre-generated drafts and the end-to-end parser’s current generation. To address this challenge, we design a two-part mechanism: (i) align drafts to the accepted tokens using a short *reference window*, and (ii) verify *multiple* candidates in one forward pass via a *prefix-tree* formulation with a specialized attention mask. Figure 2 provides an overall visualization of the core mechanism, and we formalize it in the remainder of this section.

Preliminaries. For any sequence a , the slice notation $a_{p:q}$ denotes the contiguous subsequence $(a_p, \dots, a_q)$, and $|a|$ denotes its length (in tokens).

Draft–target matching process. At decoding step t , let $\hat{\mathbf{y}}_{1:t}$ denote the accepted token sequence of the target model. Let n denote the desired window

length. As shown in Fig. 2(a), we take a reference window to be the most recent n tokens of the accepted token sequence. Thus, the reference window can be expressed as $\mathbf{w} = \hat{\mathbf{y}}_{t-n+1:t}$.

We slide the token sequence of the reference window $\mathbf{w}$ across the draft $\tilde{\mathbf{y}}$ and record every start position where it matches, forming $\mathcal{J}(\tilde{\mathbf{y}})$:

$$\mathcal{J}(\tilde{\mathbf{y}}) = \{ j \mid \tilde{\mathbf{y}}_{j:j+n-1} = \mathbf{w} \text{ and } 1 \leq j \leq |\tilde{\mathbf{y}}| - n + 1 \}, \quad (5)$$

where j indexes the start position of each match. We then extract the suffixes strictly following each matched window and collect them across all drafts in $\tilde{\mathcal{Y}}$:

$$\mathcal{C} = \left\{ \tilde{\mathbf{y}}_{j+n:|\tilde{\mathbf{y}}|} \mid \tilde{\mathbf{y}} \in \tilde{\mathcal{Y}} \text{ and } j \in \mathcal{J}(\tilde{\mathbf{y}}) \text{ and } j+n \leq |\tilde{\mathbf{y}}| \right\}. \quad (6)$$

Through the above matching and extraction steps, we obtain a set of candidate suffixes $\mathcal{C}$ for verification in the subsequent stage.

Prefix-tree batching mechanism. When the number of candidates $|\mathcal{C}| > 1$, verifying each candidate independently is redundant. To address this issue, we organize $\mathcal{C}$ into a prefix tree $\mathcal{T}$ that merges common prefixes, thereby enabling parallel verification. As shown in Fig. 2(b), each node v in $\mathcal{T}$ represents a unique prefix. Let $\pi(v)$ denote the token sequence along the path from the root to v . For example, if the node v is **parallel** (in Fig. 2(b)), then $\pi(v) = \text{tokens in parallel}$.

For any node v , we define the set of possible next tokens $\text{Next}(v)$ as

$$\text{Next}(v) = \left\{ \mathbf{c}_{|\pi(v)|+1} \mid \mathbf{c} \in \mathcal{C} \text{ and } \mathbf{c}_{1:|\pi(v)|} = \pi(v) \right\}. \quad (7)$$

Among all sequences $\mathbf{c} \in \mathcal{C}$ that share the prefix $\pi(v)$, $\text{Next}(v)$ collects the distinct tokens that appear immediately after this prefix. In Fig. 2(b), for the node corresponding to the prefix **tokens**, we have $\text{Next}(v) = \{\text{in, simultaneously, together}\}$. For each token $u \in \text{Next}(v)$, there exists a unique node w with $\pi(w) = \pi(v) \oplus u$, where $\oplus$ denotes concatenation. We record w as $\text{child}(v, u)$, and then create a directed edge from v to $\text{child}(v, u)$. In other words, each child node w encodes not only the token u itself, but also its prefix context $\pi(v)$, thereby specifying the occurrence of u within a candidate sequence.

With the above definition, we construct the prefix tree recursively. Starting from the root, which represents the empty prefix ($\pi(\text{root}) = \emptyset$), we expand newly created children until every sequence in $\mathcal{C}$ corresponds to a root-to-leaf path. The resulting tree compactly encodes all candidate continuations by sharing common prefixes as shown in Fig. 2(b).

To enable parallel verification, we linearize $\mathcal{T}$ into a packed sequence $\mathcal{P}$ and apply a *tree-ancestry attention mask*: a token at node v only attends to the accepted token sequence $\hat{\mathbf{y}}_{1:t}$ and tokens along v 's ancestor path (see Fig. 2(c)). This processes all candidate paths in one pass while preserving autoregressive conditioning.

Verification and acceptance. Let $\tau \in (0, 1)$ be the acceptance threshold. For any node v , the model produces a next-token distribution

$$p_{\theta}(\cdot \mid z, \hat{\mathbf{y}}_{1:t} \oplus \pi(v)), \quad (8)$$

where z is the current visual input (a region crop or the full page). We perform a greedy traversal of the prefix tree to obtain the final accepted token sequence.

At each step, we select the most probable next token among the candidate set $\text{Next}(s)$:

$$u^* = \arg \max_{u \in \text{Next}(s)} p_{\theta}(u \mid z, \hat{\mathbf{y}}_{1:t} \oplus \pi(s)), \quad (9)$$

where s is the current node. Let $\mathcal{V}$ denote the vocabulary. We define $\hat{u} \in \mathcal{V}$ to be the token with the highest model probability under the current context:

$$\hat{u} = \arg \max_{u \in \mathcal{V}} p_{\theta}(u \mid z, \hat{\mathbf{y}}_{1:t} \oplus \pi(s)). \quad (10)$$

We accept u^* and move to its child node if

$$\log p_{\theta}(u^* \mid z, \hat{\mathbf{y}}_{1:t} \oplus \pi(s)) - \log p_{\theta}(\hat{u} \mid z, \hat{\mathbf{y}}_{1:t} \oplus \pi(s)) \geq \log \tau. \quad (11)$$

If the condition fails or $\text{Next}(s) = \emptyset$, we stop at the current node s . Additionally, if s is a leaf node, there is no admissible next token and the traversal stops at s . Upon termination, we update the accepted token sequence:

$$\hat{\mathbf{y}}_{1:t_{\text{new}}} = \hat{\mathbf{y}}_{1:t} \oplus \pi(s) \oplus \hat{u}. \quad (12)$$

By organizing candidates into a tree and verifying them in parallel, each step can accept multiple tokens at once, substantially reducing the number of decoding steps. Tree-structured batching increases per-step compute utilization without elongating the latency-critical path, yielding significant wall-clock speedup. The decoupled design leverages the high-throughput pipeline while keeping the target model as the arbiter that corrects draft errors.

4 Experiments

4.1 Datasets

We evaluate on three public benchmarks: OmniDocBench v1.5 [30], olmOCR-Bench [33], and Ocean-OCR-Bench [7]. OmniDocBench v1.5 [30] contains 1,355 PDF pages spanning nine document types and provides rich evaluation for document parsing, including 15 block-level categories and 4 span-level elements with text, LaTeX formulas, tables, reading-order annotations, and page/block attributes. We follow the official end-to-end evaluation setup. olmOCR-Bench [33] includes 1,403 PDFs paired with 7,010 unit tests that check properties of PDF-to-Markdown conversion, such as content presence, natural reading order, table fidelity, and mathematical expressions. We report the official aggregate score under its evaluation protocol. Ocean-OCR-Bench [7] is a bilingual page-level evaluation built from 200 document images (100 English and 100 Chinese). The official metrics include normalized edit distance, F1, precision, recall, BLEU, and METEOR; we adopt the same protocol across different settings.

4.2 Metrics

For a comprehensive evaluation, we report three efficiency-related metrics: Decoding Speedup, End-to-End Speedup, and Average Acceptance Length.

Decoding Speedup [47]. Decoding Speedup measures the acceleration of the latency-critical generation loop:

$$\text{SR}_{\text{decode}} = \frac{T_{\text{decode}}^{\text{AR}}}{T_{\text{decode}}^{\text{Spec}}}. \quad (13)$$

Here, $T_{\text{decode}}^{\text{AR}}$ and $T_{\text{decode}}^{\text{Spec}}$ are wall-clock times for standard autoregressive decoding and our speculative decoder, respectively. Unless otherwise noted, T_{decode} *includes* the draft model’s forward passes used for speculation, target-model parallel verification, accept/reject control flow (including rollbacks), KV-cache maintenance, and communication overheads; it *excludes* disk I/O, non-generative preprocessing, and prefill not executed inside the decoding loop.

End-to-End Speedup [47]. To reflect user-perceived latency, we report end-to-end Speedup:

$$\text{SR}_{\text{e2e}} = \frac{T_{\text{full}}^{\text{AR}}}{T_{\text{full}}^{\text{Spec}}}, \quad (14)$$

where T_{full} measures the time from page image input to final parsing result, *including* vision/prefill computation (e.g., visual encoder and text prefill), while excluding disk I/O. Draft generation triggered specifically for speculation (e.g., per-region drafts) is counted in both $\text{SR}_{\text{decode}}$ and SR_{e2e} .

Average Acceptance Length [19]. To quantify how many decoding steps are saved by speculation, we report Average Acceptance Length (AAL). For verification step k , let α_k denote the number of consecutive draft tokens accepted by the target model before the first mismatch (full rejection gives $\alpha_k = 0$). Then,

$$\text{AAL} = \frac{1}{N} \sum_{k=1}^N \alpha_k, \quad (15)$$

with N being the number of verification steps. Larger AAL indicates more tokens skipped per step and thus higher potential speedup, though the realized SR also depends on per-step verification overhead and parallel efficiency.

4.3 Implementation Details

We validate the proposed acceleration method on several mainstream end-to-end parsers, including dots.ocr [36], HunyuanOCR [38], Qwen2.5-VL-3B [3], Qwen2.5-VL-7B [3], Qwen3-VL-2B [2], and Qwen3-VL-8B [2]. All experiments are conducted on NVIDIA A100 GPUs. For fair comparison, all methods are evaluated using the same Hugging Face Transformers stack [45], with FlexAttention [35] enabled for attention computation. Our method uses PP-StructureV3 [11] by default for layout analysis and region draft generation. In Decoupled

Table 1: Acceleration across different models on OmniDocBench v1.5. Results are reported for our proposed Hierarchical Speculative Decoding. AAL denotes the average number of accepted draft tokens per verification step; SR_{decode} is the decode-only speedup (draft generation + verification); SR_{e2e} is the end-to-end page-level latency speedup (including vision/prefill stages).

Model	Parameters	Overall		Slides		Academic Papers	Book	Textbook	Exam Papers	Magazine	Newspaper	Notes	Financial Report
		AAL	SR_{decode}	SR_{e2e}	SR_{e2e}	SR_{e2e}	SR_{e2e}	SR_{e2e}	SR_{e2e}	SR_{e2e}	SR_{e2e}	SR_{e2e}	SR_{e2e}
Qwen2.5-VL-7B [3]	8B	3.56	2.13×	2.10×	1.43×	2.35×	1.99×	1.73×	1.77×	2.17×	2.95×	1.78×	1.94×
Qwen2.5-VL-3B [3]	4B	2.52	2.14×	2.12×	1.93×	2.37×	2.02×	1.80×	1.77×	2.39×	2.80×	1.99×	2.19×
Qwen3-VL-8B [2]	9B	3.98	2.62×	2.61×	1.63×	2.56×	2.29×	2.02×	2.18×	2.59×	4.62×	1.86×	1.91×
Qwen3-VL-2B [2]	2B	3.33	2.20×	2.18×	1.35×	2.13×	1.82×	1.87×	2.04×	2.61×	4.03×	1.69×	1.75×
dots.ocr [36]	3B	3.98	2.44×	2.42×	1.52×	3.47×	2.28×	2.29×	2.04×	2.34×	2.98×	1.39×	4.89×
HunyuanOCR [38]	0.9B	4.55	2.82×	2.78×	1.58×	3.41×	4.00×	1.92×	1.72×	2.60×	4.30×	1.98×	7.04×

Table 2: Acceleration across different models on olmOCR-Bench. Results are reported for our proposed Hierarchical Speculative Decoding.

Model	Parameters	Overall		arXiv Math	Old Scans Math	Tables	Old Scans	Headers Footers	Multi Column	Long Tiny Text	
		AAL	SR_{decode}	SR_{e2e}	SR_{e2e}	SR_{e2e}	SR_{e2e}	SR_{e2e}	SR_{e2e}	SR_{e2e}	
Qwen2.5-VL-7B [3]	8B	2.47	2.05×	2.01×	2.04×	1.57×	1.49×	1.01×	2.06×	2.53×	1.59×
Qwen2.5-VL-3B [3]	4B	2.66	2.67×	2.64×	2.37×	1.78×	2.41×	1.05×	3.18×	4.07×	3.08×
Qwen3-VL-8B [2]	9B	3.08	2.42×	2.40×	2.51×	1.64×	1.60×	1.11×	2.65×	3.75×	2.19×
Qwen3-VL-2B [2]	2B	3.91	2.98×	2.97×	3.00×	1.65×	2.20×	1.46×	3.20×	4.48×	2.54×
dots.ocr [36]	3B	3.03	2.30×	2.27×	2.11×	2.10×	2.45×	1.58×	2.70×	2.11×	2.61×
HunyuanOCR [38]	0.9B	3.54	2.50×	2.46×	2.03×	1.40×	3.77×	1.28×	3.06×	3.72×	1.93×

Speculative Verification, we set the reference window length to $n = 3$, the acceptance threshold to $\tau = 0.75$, and the maximum length of the packed sequence $\mathcal{P}$ to 128 tokens.

4.4 Comprehensive Evaluations and Comparisons

Acceleration Performance Across Diverse Document Types and Models. To comprehensively evaluate the effectiveness of our hierarchical speculative decoding (HSD) approach, we conduct experiments on three benchmarks (OmniDocBench v1.5, olmOCR-Bench, and Ocean-OCR-Bench) across multiple end-to-end parsers, including specialized document parsers (dots.ocr, HunyuanOCR) and general-purpose VLMs (Qwen2.5-VL-7B/3B, Qwen3-VL-8B/2B). As shown in Tabs. 1 to 3, HSD consistently delivers positive and substantial speedups across diverse models and benchmarks with no obvious slowdown. In particular, on the state-of-the-art end-to-end parser HunyuanOCR, we obtain end-to-end speedups of $2.78\times$ (OmniDocBench v1.5), $2.46\times$ (olmOCR-Bench), and $3.29\times$ (Ocean-OCR-Bench), and observe similar trends on other parsers. The speedup magnitude varies by document type, mainly due to differences in output length, layout structure, and draft quality. In general, long documents with multiple semantic blocks (e.g., *Newspaper* and *Academic Papers* in OmniDocBench v1.5) offer higher region-level parallelism and thus larger gains, whereas challenging handwritten or degraded scans (e.g., *Old Scans* in olmOCR-Bench) often yield lower-quality drafts, reducing acceptance and limiting acceleration. We provide

Table 3: Acceleration performance across different models on Ocean-OCR-Bench. Results are reported for our proposed Hierarchical Speculative Decoding.

Model	Parameters	Overall			English Chinese	
		AAL	SR_{decode}	SR_{e2e}	SR_{e2e}	SR_{e2e}
Qwen2.5-VL-7B [3]	8B	4.91	3.03×	3.00×	3.62×	2.62×
Qwen2.5-VL-3B [3]	4B	2.63	2.78×	2.72×	4.51×	2.04×
Qwen3-VL-8B [2]	9B	6.76	3.74×	3.70×	3.54×	3.57×
Qwen3-VL-2B [2]	2B	3.11	3.02×	2.99×	4.81×	2.07×
dots.ocr [36]	3B	5.79	3.79×	3.68×	3.61×	3.75×
HunyuanOCR [38]	0.9B	4.73	3.37×	3.29×	3.96×	2.86×

Table 4: Comparisons with speculative decoding baselines under the same target model.

Target Model	Method	Venue	OmniDocBench v1.5		olmOCR-Bench		Ocean-OCR-Bench	
			AAL	SR_{e2e}	AAL	SR_{e2e}	AAL	SR_{e2e}
Qwen2.5-VL-3B	VSD [19]	ICML 2023	5.51	1.01×	5.62	1.02×	5.71	1.12×
	Spec-LLaVA [17]	ICML 2025	1.19	1.01×	1.23	1.02×	1.40	1.08×
	SpecVLM [16]	arXiv 2025	1.89	1.07×	1.79	1.23×	1.87	1.18×
	Medusa [5]	ICML 2024	0.33	1.26×	0.31	1.28×	0.35	1.32×
	MSD [24]	arXiv 2025	0.81	1.30×	1.15	1.67×	1.10	1.65×
	Dream [15]	NeurIPS 2025	2.86	1.46×	2.87	1.62×	3.10	1.53×
	HiVis [46]	CVPR 2025	1.04	1.56×	1.22	1.65×	1.44	1.72×
	Sparrow [47]	arXiv 2026	0.96	1.57×	0.99	1.53×	1.02	1.59×
	EAGLE-2 [21]	EMNLP 2024	1.50	1.69×	1.56	1.86×	1.19	1.69×
	ViSpec [18]	NeurIPS 2025	1.64	1.75×	1.62	1.90×	1.22	1.72×
	HSD	Ours	2.52	2.12×	2.66	2.64×	2.63	2.72×

qualitative visualizations in the supplementary material to substantiate these observations. Overall, HSD remains broadly effective across end-to-end parsers and document domains, demonstrating strong generality.

Comparisons with Existing Speculative Decoding Baselines. We compare HSD with representative speculative decoding methods on document parsing benchmarks under the same target model. Specifically, for vanilla speculative decoding (VSD) [19], we use a lightweight InternVL3.5-1B [42] model as the drafter for the Qwen2.5-VL-3B target model, since no smaller model from the same family is available. We further adapt Medusa [5] and EAGLE-2 [21] to VLM-based document parsing by integrating their drafting modules into the target VLM. For fairness, we also task-adapt the drafters of VSD, Medusa, EAGLE-2, and recent VLM-oriented speculative decoding methods, including Spec-LLaVA [17], SpecVLM [16], MSD [24], Dream [15], HiVis [46], Sparrow [47], and ViSpec [18], using the same document parsing data.

As shown in Tab. 4, existing speculative decoding methods provide only limited acceleration on document parsing, with the best baseline, ViSpec, achieving a speedup of 1.72–1.90×. This suggests that speculative decoding for document parsing poses a unique challenge. In contrast, HSD achieves substantially higher speedups without additional training, reaching 2.12–2.72× under the same Qwen2.5-VL-3B target model. This advantage comes from two designs tailored to document parsing: *a pipeline parser that efficiently produces high-acceptance*

Table 5: Comparison with pipeline, hybrid, and end-to-end document parsers on OmniDocBench v1.5, including end-to-end parsers with and without our HSD.

Model	Pipeline		Hybrid		End-to-End		HSD	
	MinerU2-pipeline [31]	PP-Structure V3 [11]	MonkeyOCR-pro-3B [23]	MinerU 2.5 [30]	dots. ocr [36]	Hunyuan OCR [38]	dots. ocr [36]	Hunyuan OCR [38]
Accuracy↑	75.51	86.73	88.85	90.67	86.73	94.10	88.81	94.02
Mean latency (s/sample)↓	1.79	1.99	26.69	46.74	60.06	30.47	24.82	10.96

Table 6: Acceleration when combining our hierarchical speculative decoding (HSD) with the visual token compression (VTC) method.

Method	OmniDocBench v1.5		olmOCR-Bench		Ocean-OCR-Bench	
	AAL	SR_{decode}	SR_{e2e}	AAL	SR_{decode}	SR_{e2e}
VTC (DeepSeek-OCR [44])	-	1.00×	1.00×	-	1.00×	1.00×
+ HSD	3.72	1.51×	1.56×	3.65	1.39×	1.41×

drafts, and a hierarchical framework that enables region-level parallel verification while preserving page-level parsing accuracy.

Comparisons with Existing Document Parsing Methods. We further compare end-to-end parsers [36, 38] with and without HSD against representative pipeline-based [11, 31] and hybrid [23, 30] document parsing approaches. The comparison is intended to characterize the accuracy–latency trade-off of different parsing paradigms under their standard inference settings, rather than to perform a latency-matched evaluation. As shown in Tab. 5, pure pipelines are fast but often result in lower output quality, while hybrid systems aim to strike a balance between quality and latency. End-to-end parsers can achieve stronger quality on complex pages but are markedly slower. By integrating HSD, we reduce the latency of end-to-end parsers by 2.42–2.78× with essentially unchanged quality, bringing them closer to the efficiency level of hybrid systems. Overall, HSD establishes a new paradigm for document parsing, enabling end-to-end parsers to attain hybrid-level efficiency without altering the underlying models or introducing extra training.

Combining with the Visual Token Compression Method. Complementary to HSD, visual token compression (VTC) accelerates document parsing by reducing the number of visual tokens and thus the attention cost (e.g., DeepSeek-OCR [44]). To examine whether the two directions are compatible, we integrate HSD with DeepSeek-OCR. As shown in Tab. 6, HSD brings additional speedups on top of VTC, suggesting that our method is plug-and-play and stackable with other acceleration techniques.

4.5 Ablation Study

Accuracy Analysis of Hierarchical Design. Tab. 7 validates the effectiveness of our hierarchical design and demonstrates near-lossless acceleration. Using only Stage 1 causes significant performance degradation—dots.ocr drops from 88.41 to 70.47 on OmniDocBench v1.5—due to the lack of global context and

Table 7: Accuracy comparison of the draft model, baseline parser, and our hierarchically accelerated variants.

Method	OmniDocBench v1.5	olmOCR-Bench	Ocean-OCR-Bench
Pipeline (Draft)	86.73	65.80	85.20
dots.ocr			
Baseline	88.41	79.90	91.45
Stage 1 only	70.47	67.30	86.92
Stage 1+2 (Ours)	88.81	79.40	92.56

Table 8: Ablation of our design on OmniDocBench v1.5 with dots.ocr.

Method	AAL	SR_{decode}	SR_{e2e}
Baseline	-	1.00×	1.00×
+ Page-level Spec. Decoding only	2.49	2.11×	2.09×
+ Hierarchical Spec. Decoding	3.98	2.44×	2.42×

inherited layout segmentation errors from cropped inputs. Thus, Stage 2 is essential for recovering the Stage-1-degraded accuracy to the baseline level (88.81 on OmniDocBench v1.5, 92.56 on Ocean-OCR-Bench). Despite tolerance-based draft verification, the final accuracy remains comparable to the baseline across benchmarks. These results show that HSD achieves substantial speedups while maintaining parsing quality.

Impact of Framework Designs. Tab. 8 ablates the contribution of our framework design on OmniDocBench v1.5 with dots.ocr as the target parser. Compared with the autoregressive baseline, using a fast pipeline drafter already improves the end-to-end speedup to 2.09×, demonstrating that an efficient drafting stage provides a solid foundation for acceleration. By introducing the hierarchical two-stage design, our method further increases the end-to-end speedup from 2.09× to 2.42× and improves AAL from 2.49 to 3.98, benefiting from region-level parallel verification in Stage 1. This progression from the baseline to the complete framework demonstrates that the hierarchical design and speculative verification work synergistically to achieve optimal acceleration.

Impacts of Drafters. In our experiments, we primarily use PP-StructureV3 as the drafter, but our method is flexible and can accommodate other pipelines as drafters. To evaluate the impact of different drafters, we test with MinerU2-pipeline [31] and Docling [27]. Additionally, to mimic different real-world drafter behaviors, we add various types of noise to drafts generated by PP-StructureV3, including image degradation, layout detection noise, and recognition errors. As shown in Fig. 3, despite reduced draft quality, HSD maintains a speedup of over 2.3×, demonstrating its plug-and-play nature and compatibility with various pipelines. This highlights the practical value and robustness of our approach.

Impacts of Hyperparameters. We further investigate the choices of the acceptance threshold and the reference window size using dots.ocr. As shown in Fig. 4, the acceleration improves as the acceptance threshold decreases. However, when $\tau < 0.75$, the parsing accuracy drops noticeably, while $\tau = 0.75$ achieves the best trade-off between acceleration and accuracy. In addition, the acceleration performance is relatively robust to the reference window size, with

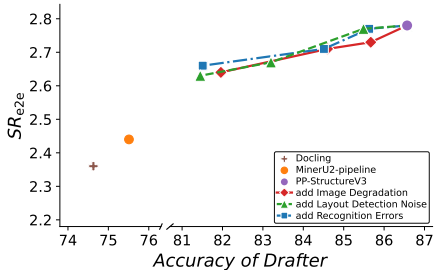


Fig. 3: Impacts of using different pipelines as drafters. *Accuracy of Drafter* is the OmniDocBench v1.5 score; “add ...” denotes noise injected into PP-StructureV3 drafts.

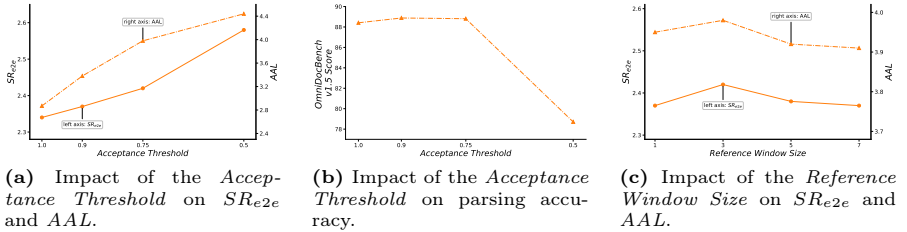


Fig. 4: Impacts of hyperparameter settings on acceleration and parsing accuracy.

$n = 3$ yielding the best overall results. Therefore, we set the acceptance threshold $\tau = 0.75$ and the reference window size $n = 3$.

5 Conclusion

In this work, we propose a hierarchical speculative decoding paradigm to address the inference-speed bottleneck of VLM-based end-to-end document parsers. By dividing long, structured document parsing outputs into multiple regions, we enable region-level parallel speculative verification. Building on the refined region-level outputs, we perform a full-page speculative verification to preserve the global coherence and correct the remaining errors. To further improve the efficiency of HSD, we propose decoupled speculative verification, which resolves draft-target misalignment and enables efficient verification over multiple candidates. Extensive experiments demonstrate that our approach achieves significant near-lossless speedups: $2.78\times$ acceleration on OmniDocBench v1.5 with HunyuanOCR and up to $7.04\times$ acceleration on long-document parsing tasks. These results highlight the practicality and generalizability of our method, offering a plug-and-play solution to accelerate VLM-based document parsers without architectural changes or retraining. We hope that the concept of hierarchical verification will inspire further research in speculative decoding across other fields.

Acknowledgement

This research is supported in part by National Natural Science Foundation of China (Grant No. 62476093), the Natural Science Foundation of Guangdong Province (Grant No. 2026A1515012038), Shanghai Artificial Intelligence Laboratory, Guangdong Basic and Applied Basic Research Foundation (Grant No. 2026A1515011967) and Shenzhen Fundamental Research (Key Program) under Grant JCYJ20241202124931042.

References

1. Alayrac, J.B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al.: Flamingo: a visual language model for few-shot learning. NeurIPS (2022)

2. Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., Ge, W., Guo, Z., Huang, Q., Huang, J., Huang, F., Hui, B., Jiang, S., Li, Z., Li, M., Li, M., Li, K., Lin, Z., Lin, J., Liu, X., Liu, J., Liu, C., Liu, Y., Liu, D., Liu, S., Lu, D., Luo, R., Lv, C., Men, R., Meng, L., Ren, X., Ren, X., Song, S., Sun, Y., Tang, J., Tu, J., Wan, J., Wang, P., Wang, P., Wang, Q., Wang, Y., Xie, T., Xu, Y., Xu, H., Xu, J., Yang, Z., Yang, M., Yang, J., Yang, A., Yu, B., Zhang, F., Zhang, H., Zhang, X., Zheng, B., Zhong, H., Zhou, J., Zhou, F., Zhou, J., Zhu, Y., Zhu, K.: Qwen3-VL technical report. arXiv preprint arXiv:2511.21631 (2025)
3. Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., Zhong, H., Zhu, Y., Yang, M., Li, Z., Wan, J., Wang, P., Ding, W., Fu, Z., Xu, Y., Ye, J., Zhang, X., Xie, T., Cheng, Z., Zhang, H., Yang, Z., Xu, H., Lin, J.: Qwen2.5-VL Technical Report. arXiv preprint arXiv:2502.13923 (2025)
4. Blecher, L., Cucurull, G., Scialom, T., Stojnic, R.: Nougat: Neural optical understanding for academic documents. arXiv preprint arXiv:2308.13418 (2023)
5. Cai, T., Li, Y., Geng, Z., Peng, H., Lee, J.D., Chen, D., Dao, T.: Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In: ICML. pp. 5209–5235 (2024)
6. Chen, C., Borgeaud, S., Irving, G., Lespiau, J.B., Sifre, L., Jumper, J.: Accelerating large language model decoding with speculative sampling. arXiv preprint arXiv:2302.01318 (2023)
7. Chen, S., Guo, X., Li, Y., Zhang, T., Lin, M., Kuang, D., Zhang, Y., Ming, L., Zhang, F., Wang, Y., et al.: Ocean-OCR: Towards general OCR application via a vision-language model. arXiv preprint arXiv:2501.15558 (2025)
8. Chen, X., Wang, X., Changpinyo, S., Piergiovanni, A., Padlewski, P., Salz, D., Goodman, S., Grycner, A., Mustafa, B., Beyer, L., et al.: PaLI: A jointly-scaled multilingual language-image model. In: ICLR (2022)
9. chatdoc com: Ocrflux. <https://github.com/chatdoc-com/OCRFlux> (2025), accessed:2025-11-10
10. Cui, C., Sun, T., Liang, S., Gao, T., Zhang, Z., Liu, J., Wang, X., Zhou, C., Liu, H., Lin, M., Zhang, Y., Zhang, Y., Zheng, H., Zhang, J., Zhang, J., Liu, Y., Yu, D., Ma, Y.: PaddleOCR-VL: Boosting multilingual document parsing via a 0.9B ultra-compact vision-language model. arXiv preprint arXiv:2510.14528 (2025)
11. Cui, C., Sun, T., Lin, M., Gao, T., Zhang, Y., Liu, J., Wang, X., Zhang, Z., Zhou, C., Liu, H., et al.: PaddleOCR 3.0 technical report. arXiv preprint arXiv:2507.05595 (2025)
12. Elhoushi, M., Shrivastava, A., Liskovich, D., Hosmer, B., Wasti, B., Lai, L., Mahmoud, A., Acun, B., Agarwal, S., Roman, A., Aly, A., Chen, B., Wu, C.J.: LayerSkip: Enabling early exit inference and self-speculative decoding. In: ACL. pp. 12622–12642 (2024)
13. Feng, H., Wei, S., Fei, X., Shi, W., Han, Y., Liao, L., Lu, J., Wu, B., Liu, Q., Lin, C., et al.: Dolphin: Document image parsing via heterogeneous anchor prompting. arXiv preprint arXiv:2505.14059 (2025)
14. Gagrani, M., Goel, R., Jeon, W., Park, J., Lee, M., Lott, C.: On speculative decoding for multimodal large language models. In: CVPR Workshops. pp. 8285–8289 (2024)
15. Hu, Y., Xia, T., Liu, Z., Raman, R., Liu, X., BAO, B., Sather, E., Thangarasa, V., Zhang, S.Q.: DREAM: Drafting with refined target features and entropy-adaptive cross-attention fusion for multimodal speculative decoding. In: NeurIPS. pp. 167592–167612 (2025)

16. Huang, H., Yang, F., Liu, Z., Yin, X., Li, D., Ren, P., Barsoum, E.: SpecVLM: Fast speculative decoding in vision-language models. arXiv preprint arXiv:2509.11815 (2025)
17. Huo, M., Zhang, J., Wang, H., Xu, J., Chen, Z., Tai, H., Chen, I.Y.: Spec-LLaVA: Accelerating vision-language models with dynamic tree-based speculative decoding. In: ICML Workshops (2025)
18. Kang, J., Shu, H., Li, W., Zhai, Y., Chen, X.: ViSpec: Accelerating vision-language models with vision-aware speculative decoding. In: NeurIPS. pp. 115511–115532 (2025)
19. Leviathan, Y., Kalman, M., Matias, Y.: Fast inference from transformers via speculative decoding. In: ICML. pp. 19274–19286 (2023)
20. Li, J., Li, D., Xiong, C., Hoi, S.: BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In: ICML. pp. 12888–12900 (2022)
21. Li, Y., Wei, F., Zhang, C., Zhang, H.: EAGLE-2: Faster inference of language models with dynamic draft trees. In: EMNLP. pp. 7421–7432 (2024)
22. Li, Y., Wei, F., Zhang, C., Zhang, H.: EAGLE: Speculative sampling requires rethinking feature uncertainty. In: ICML. pp. 28935–28948 (2024)
23. Li, Z., Liu, Y., Liu, Q., Ma, Z., Zhang, Z., Zhang, S., Guo, Z., Zhang, J., Wang, X., Bai, X.: MonkeyOCR: Document parsing with a structure-recognition-relation triplet paradigm. arXiv preprint arXiv:2506.05218 (2025)
24. Lin, L., Lin, Z., Zeng, Z., Ji, R.: Speculative decoding reimaged for multimodal large language models. arXiv preprint arXiv:2505.14260 (2025)
25. Liu, H., Li, C., Li, Y., Lee, Y.J.: Improved baselines with visual instruction tuning. arXiv:2310.03744 (2023)
26. Liu, Y., Zhao, Z., Tian, L., Wang, H., Ye, X., You, Y., Yu, Z., Wu, C., Zhou, X., Yu, Y., et al.: POINTS-Reader: Distillation-free adaptation of vision-language models for document conversion. arXiv preprint arXiv:2509.01215 (2025)
27. Livathinos, N., Auer, C., Lysak, M., Nassar, A., Dolfi, M., Vagenas, P., Ramis, C.B., Omenetti, M., Dinkla, K., Kim, Y., et al.: Docling: An efficient open-source toolkit for AI-driven document conversion. arXiv preprint arXiv:2501.17887 (2025)
28. Mandalm, S.: Nanonets-ocr-s. <https://nanonets.com/research/nanonets-ocr-s/> (2025), accessed:2025-11-10
29. Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Zhang, Z., Wong, R.Y.Y., Zhu, A., Yang, L., Shi, X., Shi, C., Chen, Z., Arfeen, D., Abhyankar, R., Jia, Z.: SpecInfer: Accelerating large language model serving with tree-based speculative inference and verification. In: ASPLOS. p. 932–949 (2024)
30. Niu, J., Liu, Z., Gu, Z., Wang, B., Ouyang, L., Zhao, Z., Chu, T., He, T., Wu, F., Zhang, Q., Jin, Z., Liang, G., Zhang, R., Zhang, W., Qu, Y., Ren, Z., Sun, Y., Zheng, Y., Ma, D., Tang, Z., Niu, B., Miao, Z., Dong, H., Qian, S., Zhang, J., Chen, J., Wang, F., Zhao, X., Wei, L., Li, W., Wang, S., Xu, R., Cao, Y., Chen, L., Wu, Q., Gu, H., Lu, L., Wang, K., Lin, D., Shen, G., Zhou, X., Zhang, L., Zang, Y., Dong, X., Wang, J., Zhang, B., Bai, L., Chu, P., Li, W., Wu, J., Wu, L., Li, Z., Wang, G., Tu, Z., Xu, C., Chen, K., Qiao, Y., Zhou, B., Lin, D., Zhang, W., He, C.: MinerU2.5: A decoupled vision-language model for efficient high-resolution document parsing. arXiv preprint arXiv:2509.22186 (2025)
31. OpenDataLab: MinerU. <https://github.com/opendatalab/MinerU> (2025), accessed 2026-03-04
32. Paruchuri, V.: Marker. <https://github.com/datalab-to/marker> (2025), accessed:2025-11-10

33. Poznanski, J., Rangapur, A., Borchardt, J., Dunkelberger, J., Huff, R., Lin, D., Wilhelm, C., Lo, K., Soldaini, L.: olmOCR: Unlocking trillions of tokens in PDFs with vision language models. arXiv preprint arXiv:2502.18443 (2025)
34. Poznanski, J., Soldaini, L., Lo, K.: olmOCR 2: Unit test rewards for document OCR. arXiv preprint arXiv:2510.19817 (2025)
35. PyTorch, T.: FlexAttention: The flexibility of pytorch with the performance of flashattention. <https://pytorch.org/blog/flexattention/> (2025), accessed:2025-11-10
36. rednote: dots.ocr: Multilingual document layout parsing in a single vision-language model. <https://github.com/rednote-hilab/dots.ocr> (2025), accessed:2025-11-10
37. Sun, X., Ge, T., Wei, F., Wang, H.: Instantaneous grammatical error correction with shallow aggressive decoding. arXiv preprint arXiv:2106.04970 (2021)
38. Team, H.V., Lyu, P., Wan, X., Li, G., Peng, S., Wang, W., Wu, L., Shen, H., Zhou, Y., Tang, C., Yang, Q., Peng, Q., Luo, B., Yang, H., Zhang, X., Zhang, J., Peng, H., Yang, H., Xie, S., Zhou, L., Pei, G., Wu, B., Yan, R., Wu, K., Yang, J., Wang, B., Liu, K., Zhu, J., Jiang, J., Linus, Hu, H., Zhang, C.: HunyuanOCR technical report. arXiv preprint arXiv:2511.19575 (2025)
39. Team, M.A.: Mistral-ocr. https://mistral.ai/news/mistral-ocr?utm_source=ai-bot.cn (2025), accessed:2025-11-10
40. Wang, B., Wu, B., Li, W., Fang, M., Huang, Z., Huang, J., Wang, H., Liang, Y., Chen, L., Chu, W., Qi, Y.: Infinity Parser: Layout aware reinforcement learning for scanned document parsing. arXiv preprint arXiv:2506.03197 (2025)
41. Wang, B., Xu, C., Zhao, X., Ouyang, L., Wu, F., Zhao, Z., Xu, R., Liu, K., Qu, Y., Shang, F., et al.: MinerU: An open-source solution for precise document content extraction. arXiv preprint arXiv:2409.18839 (2024)
42. Wang, W., Gao, Z., Gu, L., Pu, H., Cui, L., Wei, X., Liu, Z., Jing, L., Ye, S., Shao, J., Wang, Z., Chen, Z., Zhang, H., Yang, G., Wang, H., Wei, Q., Yin, J., Li, W., Cui, E., Chen, G., Ding, Z., Tian, C., Wu, Z., Xie, J., Li, Z., Yang, B., Duan, Y., Wang, X., Hou, Z., Hao, H., Zhang, T., Li, S., Zhao, X., Duan, H., Deng, N., Fu, B., He, Y., Wang, Y., He, C., Shi, B., He, J., Xiong, Y., Lv, H., Wu, L., Shao, W., Zhang, K., Deng, H., Qi, B., Ge, J., Guo, Q., Zhang, W., Zhang, S., Cao, M., Lin, J., Tang, K., Gao, J., Huang, H., Gu, Y., Lyu, C., Tang, H., Wang, R., Lv, H., Ouyang, W., Wang, L., Dou, M., Zhu, X., Lu, T., Lin, D., Dai, J., Su, W., Zhou, B., Chen, K., Qiao, Y., Wang, W., Luo, G.: InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. arXiv preprint arXiv:2508.18265 (2025)
43. Wei, H., Liu, C., Chen, J., Wang, J., Kong, L., Xu, Y., Ge, Z., Zhao, L., Sun, J., Peng, Y., et al.: General OCR Theory: Towards OCR-2.0 via a unified end-to-end model. arXiv preprint arXiv:2409.01704 (2024)
44. Wei, H., Sun, Y., Li, Y.: DeepSeek-OCR: Contexts optical compression. arXiv preprint arXiv:2510.18234 (2025)
45. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., Rush, A.: Transformers: State-of-the-art natural language processing. In: ACL. pp. 38–45 (Oct 2020)
46. Xie, Z., Wang, P., Qiu, S., Cheng, J.: HiVis: Hiding visual tokens from the drafter for speculative decoding in vision-language models. In: CVPR Findings. pp. 8952–8961 (2026)

47. Zhang, L., Zhang, Z., Hong, W., Qiao, P., Li, D.: Sparrow: Text-anchored window attention with visual-semantic glimpsing for speculative decoding in video llms. arXiv preprint arXiv:2602.15318 (2026)
48. Zhang, P., Zhang, J., Cao, J., Li, H., Jin, L.: Smaller But Better: Unifying Layout Generation with Smaller Large Language Models. *IJCV* **133**, 3891–3917 (2025)
49. Zhang, Q., Wang, B., Huang, V.S.J., Zhang, J., Wang, Z., Liang, H., He, C., Zhang, W.: Document parsing unveiled: Techniques, challenges, and prospects for structured information extraction. arXiv preprint arXiv:2410.21169 (2024)
50. Zhao, W., Huang, Y., Han, X., Xiao, C., Liu, Z., Sun, M.: Ouroboros: Speculative decoding with large model enhanced drafting. arXiv preprint arXiv:2402.13720 (2024)