HPD-Parsing: Hierarchical Parallel Document Parsing

Shu Wei*, Jingjing Wu*, Lingshu Zhang*, Qunyi Xie†, Hao Zou, Le Xiang, Xu Fan, Yangliu Xu, Manhui Lin, Xiaolong Ma, Cheng Cui, Tengyu Du, YY

***Equal contribution, †Project Leader**

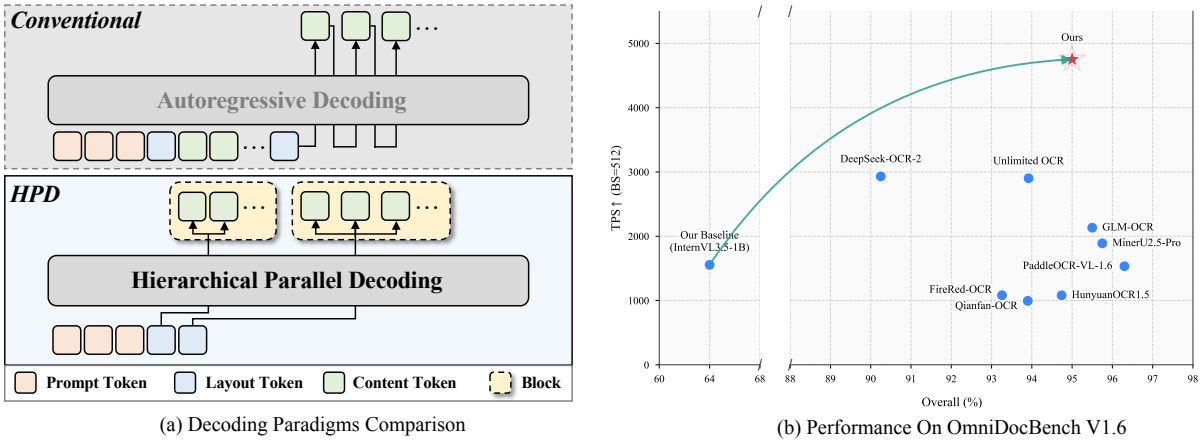
https://github.com/PaddlePaddle/PaddleOCR

https://huggingface.co/PaddlePaddle

paddleocr@baidu.com

Abstract

Efficient teamwork typically combines global coordination with parallel execution, a principle not yet fully reflected in unified Vision-Language Model (VLM)-based document parsers. Existing unified parsers process an entire page jointly but generate its output through a single token-by-token autoregressive trajectory, creating a sequential bottleneck that grows with document length. Such full-page sequential generation overlooks a key property of document parsing: layout must be analyzed globally, whereas block content can be parsed in parallel. Based on this observation, we introduce **HPD-Parsing**, which replaces full-page autoregressive generation with a **H**ierarchical **P**arallel **D**ecoding paradigm. A main layout branch organizes the overall document structure and dynamically assigns block-level content decoding to concurrent branches, while progressive multi-token prediction (P-MTP) further reduces the decoding steps within each branch. Experiments on public benchmarks show that HPD-Parsing achieves 4,752 tokens per second, delivering 2.62× the throughput of the fastest existing document parsing model and 3.06× that of the vanilla autoregressive baseline, while maintaining competitive parsing accuracy. These results establish hierarchical parallel decoding as an effective alternative to full-page autoregressive generation, opening a new direction for efficient unified document parsing.



(a) Decoding Paradigms Comparison

(b) Performance On OmniDocBench V1.6

Figure 1 | Comparison of decoding paradigms and inference efficiency. (a) Conventional autoregressive decoding generates the full-page output sequentially, whereas HPD-Parsing combines layout-coordinated parallel decoding across document blocks with P-MTP. (b) HPD-Parsing delivers a 3.06× throughput improvement over the autoregressive baseline, while achieving state-of-the-art inference efficiency with competitive parsing accuracy.

1. Introduction

Document parsing serves as a fundamental component in a wide range of downstream applications, including information extraction, document retrieval, and Retrieval-Augmented Generation (RAG), placing high demands on parsing accuracy. Compared with traditional pipeline-based systems, recent Vision-Language Model (VLM)-based parsers have substantially improved accuracy and generality by jointly modeling diverse document elements within a unified framework. However, document pages often contain dense content and require long structured outputs, causing VLM-based parsers to incur substantially higher decoding costs than traditional systems because of token-by-token autoregressive generation. As document workloads continue to grow, improving inference efficiency while preserving parsing quality has therefore become an increasingly important problem.

Existing VLM-based document parsers can be broadly categorized into pipeline-based and unified methods. Pipeline-based methods [1, 2, 3, 4, 5, 6, 7, 8, 9] typically combine specialized components for layout analysis and VLM-based content decoding, enabling flexible region-wise processing with inherently fragmented workflows. Unified methods [10, 11, 12, 13, 9, 14, 15, 16, 17, 18, 19], by contrast, integrate layout parsing and content recognition into a unified sequence generation framework, enabling joint optimization within a single model. Nevertheless, VLM-based decoding in existing parsers is still predominantly autoregressive, requiring document content to be generated token by token. This serial generation process leads to high inference latency and limited throughput, making it difficult to satisfy the efficiency requirements of large-scale document parsing.

To alleviate the inference bottleneck, recent studies have pursued two complementary directions. One line of work reduces attention cost by shortening the effective context. DeepSeek-OCR [17, 18] compresses visual tokens to reduce redundancy in high-resolution document inputs, while Unlimited OCR [20] introduces Reference Sliding Window Attention (R-SWA) to bound the textual KV cache during long-sequence decoding. A second line of work shortens the generation path by increasing decoding parallelism. Youtu-Parsing [6] introduces query- and token-level parallelism for region-wise recognition, GLM-OCR [9] employs Multi-Token Prediction (MTP) to advance multiple tokens per decoding step, and HunyuanOCR-1.5 [21] adapts DFlash for parallel draft generation. These advances offer complementary efficiency improvements, while the broader decoding paradigm for full-page document generation remains underexplored.

Beyond existing acceleration strategies, we observe a fundamental mismatch in unified document parsing: while page-level parsing requires global coordination, content decoding is largely localized. The global layout determines the spatial structure, region relationships, and reading order of a document, whereas the detailed content of each region is primarily grounded in its corresponding visual evidence and exhibits limited dependence on distant regions. This makes a single full-page autoregressive trajectory unnecessarily sequential. Moreover, the strong visual grounding of localized content yields relatively predictable generation trajectories, making document parsing well suited to multi-token prediction. Motivated by this insight, we propose HPD-Parsing, a high-throughput document parser built on Hierarchical Parallel Decoding. A main layout branch performs global coordination and dynamically decomposes the conventional single decoding trajectory into concurrent local content branches with shared-prefix KV cache reuse. Within each branch, P-MTP [22] further reduces decoding steps by predicting multiple future tokens at each iteration. Collectively, these designs shorten the effective sequential decoding path across branches and within each branch while preserving a unified generation framework. To maintain parsing accuracy during this paradigm transition,

we further adopt staged adaptation and automated difficulty-aware data curation. As shown in Figure 1(b), HPD-Parsing achieves best throughout while maintaining competitive parsing accuracy.

In summary, the key contributions of HPD-Parsing are as follows:

- **Hierarchical Parallel Decoding for High-Throughput Document Parsing.** We introduce Hierarchical Parallel Decoding (HPD), a new decoding paradigm that restructures full-page autoregressive generation into globally coordinated, localized parallel decoding. A main layout branch performs global coordination and dynamically decomposes the conventional single decoding trajectory into concurrent content branches, each responsible for a localized document region. Within each branch, P-MTP further reduces the number of decoding steps by predicting multiple future tokens at each iteration. Together with shared-prefix KV cache reuse, HPD substantially shortens the effective sequential decoding path along both branch and token dimensions.
- **Staged Adaptation with Automated Difficulty-Aware Data Curation.** We develop a staged adaptation strategy that transfers conventional autoregressive document parsing capabilities to the proposed hierarchical parallel decoding paradigm while preserving parsing accuracy. The strategy is supported by an automated difficulty-aware data curation pipeline that integrates large-scale data collection, model-assisted annotation, difficulty estimation, and balanced sampling. By progressively adapting the model and emphasizing challenging samples, the training framework mitigates the accuracy degradation caused by the transition to parallel decoding with minimal manual annotation effort.
- **State-of-the-Art Throughput with Competitive Parsing Accuracy.** HPD-Parsing achieves state-of-the-art inference efficiency on OmniDocBench v1.6, reaching a peak throughput of **4,752 Tokens Per Second (TPS)**. It delivers 1.62× the throughput of the fastest existing document parsing model and more than 3.06× that of its autoregressive baseline, while maintaining competitive parsing accuracy. These results demonstrate that document parsing can be effectively executed through global layout coordination and localized parallel decoding rather than a single sequential generation trajectory.

2. Related Work

2.1. VLM-based Document Parsing

Existing VLM-based document parsers can be broadly categorized into pipeline-based and unified methods according to their system organization. Pipeline-based methods [1, 2, 3, 4, 5, 6, 7, 8, 9] decompose document parsing into a sequence of specialized stages, typically including page-level layout analysis followed by VLM-based recognition or content decoding for localized regions. This modular design enables flexible optimization for different document elements and has demonstrated strong parsing performance in systems such as the PaddleOCR-VL and MinerU series. However, coordinating multiple components also increases system complexity and may propagate localization errors to subsequent recognition stages.

Unified methods [10, 11, 12, 13, 9, 14, 15, 16, 17, 18, 19, 20, 21] instead formulate layout parsing, reading-order prediction, and content recognition within a unified sequence generation framework. Recent studies have improved this paradigm through large-scale data curation [23, 13, 10, 11], multi-task supervision and output formatting [12], and reinforcement learning [16, 10, 19]. Despite substantial progress in parsing accuracy, VLM-based decoders in both paradigms remain predominantly autoregressive. Whether applied to localized regions or an entire page,

document content is generally generated token by token, resulting in substantial inference latency for text-dense documents with long structured outputs.

2.2. Acceleration Strategies for Document Parsing

Recent studies have accelerated VLM-based document parsing along two complementary dimensions: reducing per-step computation and shortening the sequential decoding path. DeepSeek-OCR [17, 18] compresses visual tokens to reduce the attention cost of high-resolution document inputs, while Unlimited OCR [20] introduces Reference Sliding Window Attention (R-SWA) to maintain a bounded textual KV cache during long-sequence generation. Both approaches improve efficiency by limiting the effective context processed at each decoding step. Complementary efforts increase parallelism within the decoding process to reduce the number of sequential decoding rounds. Youtu-Parsing [6] introduces query- and token-level parallelism for region-wise content recognition, while GLM-OCR [9] employs Multi-Token Prediction (MTP) to advance multiple tokens per decoding step. HunyuanOCR-1.5 [21] further adapts DFlash to OCR decoding, enabling parallel draft generation for faster autoregressive inference. Collectively, these methods reduce the cost of context processing or accelerate token generation within the conventional autoregressive pipeline. Yet, how to restructure full-page document generation beyond the conventional autoregressive trajectory remains largely unexplored.

3. Methodology

3.1. From Sequential Bottleneck to Hierarchical Parallel Decoding

Before introducing HPD-Parsing, we first analyze the inference bottleneck of standard autoregressive document parsing. For this purpose, we train a baseline document parser using InternVL3.5-1B as the backbone. The model follows the conventional unified VLM-based parsing pipeline: a document image is first encoded into visual tokens, after which the complete parsing result is generated token by token by the LLM decoder. As shown in Figure 2, we profile the encoder and decoder latency across samples grouped by output length using vLLM with an inference batch size of 16.

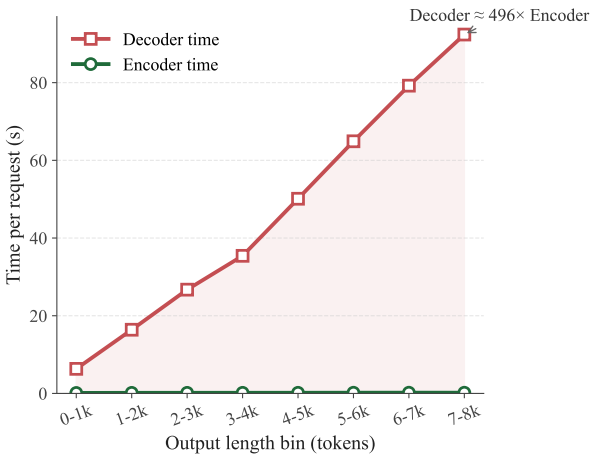


Figure 2 | Profiling results on a standard InternVL3.5-1B-based autoregressive parsing model show that decoder latency increases rapidly with output length and becomes the dominant inference cost.

As the output length increases, encoder latency remains relatively stable despite moderate variation in the number of visual tokens. Decoder latency, by contrast, grows rapidly with sequence length and progressively dominates the overall inference cost. For samples with long outputs, decoding can take nearly 500 times longer than visual encoding. These results indicate that the primary efficiency bottleneck of unified document parsing lies not in full-page visual processing, but in the long sequential execution path imposed by token-by-token autoregressive decoding.

Overcoming this bottleneck requires reconsidering the organization of document generation. Although document parsing relies on global coordination to determine spatial structure, region relationships, and reading order, the detailed content of each region is largely localized. Text paragraphs, tables, and formulas are primarily grounded in their corresponding visual regions and typically exhibit limited dependence on the detailed content of distant regions. Therefore, global document structure must be coordinated jointly, but regional contents need not be generated sequentially along a single autoregressive trajectory.

This observation motivates a hierarchical decoding organization. A main layout branch establishes the global layout structure and reading order, while dynamically decomposing the conventional single decoding trajectory into concurrent local content branches. Each content branch decodes a localized region, with shared-prefix KV cache reuse avoiding redundant computation. Within each branch, strong grounding in local visual evidence further makes the generation trajectory more predictable, creating favorable conditions for Progressive Multi-Token Prediction (P-MTP) to reduce the number of autoregressive decoding steps. Based on these insights, we formulate document parsing as Hierarchical Parallel Decoding: global layout coordination enables localized parallel decoding across content branches, while P-MTP further accelerates generation within each branch.

3.2. Overview

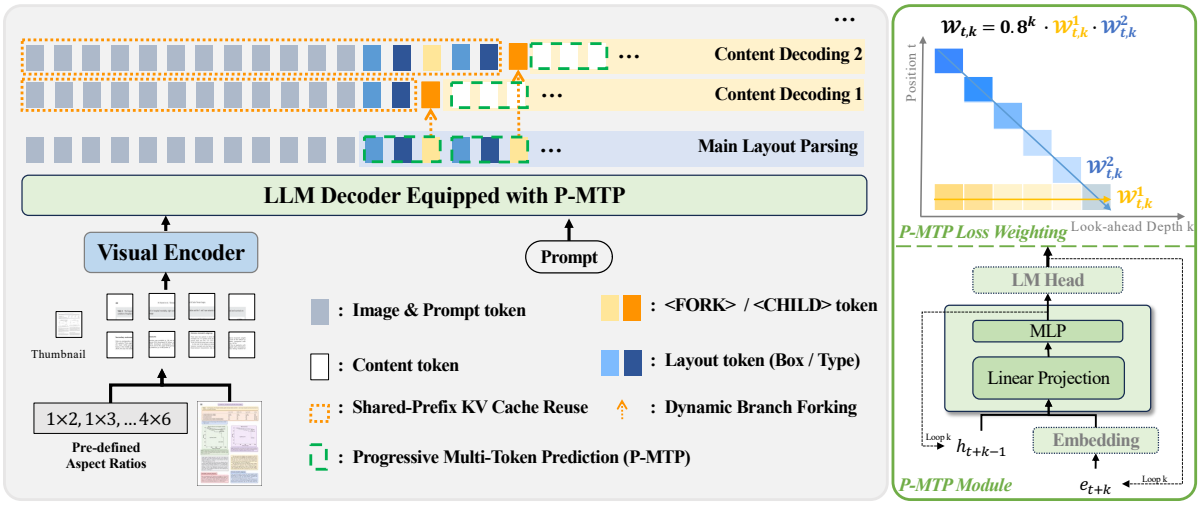
As illustrated in Figure 3, HPD-Parsing adopts InternVL3.5-1B [24] as its backbone, comprising a 0.3B InternViT visual encoder and a 0.8B LLM decoder. The visual encoder extracts high-resolution document representations, which are projected into the language space and processed by the decoder for unified document parsing.

Visual Encoder. InternViT adopts dynamic tile-based cropping to efficiently preserve high-resolution details. During both training and inference, each input image is adaptively partitioned into up to 24 tiles according to its resolution and aspect ratio. Each tile is resized to 448×448 and independently encoded, after which the resulting visual tokens are fed into the LLM.

LLM Decoder. LLM Decoder is adapted from the Qwen3-0.6B architecture, constituting approximately 0.8B parameters of the overall framework. It consists of 28 Transformer layers with a hidden dimension of 1024 and an intermediate FFN size of 3072. To manage attention complexity, the decoder utilizes Grouped-Query Attention (GQA) configured with 16 query heads and 8 key-value heads. SwiGLU is adopted as the activation function coupled with RMSNorm. Built upon this decoder, HPD-Parsing replaces the conventional single autoregressive trajectory with layout-coordinated dynamic branch forking, enabling the main layout branch and localized content branches to decode concurrently. Our previously proposed P-MTP is further integrated into each branch to reduce the remaining autoregressive decoding steps. The layout-coordinated branch decoding mechanism and its integration with P-MTP are described in Section 3.3 and Section 3.4, respectively.

3.3. Layout-Coordinated Parallel Decoding

HPD-Parsing introduces layout-coordinated parallel decoding to exploit the structural locality of document pages and shorten the sequential execution path of full-page generation. Rather than generating the complete parsing result along a single autoregressive trajectory, it assigns different decoding roles to global layout coordination and localized content generation. A main layout branch sequentially establishes the document structure in reading order, while content branches are dynamically forked to decode individual layout regions concurrently.



(a) Hierarchical Parallel Decoding Paradigm (b) P-MTP

Figure 3 | HPD-Parsing accelerates document parsing through a hierarchical parallel decoding paradigm that combines layout-coordinated branch decoding with P-MTP. In (a), the input document image is encoded into visual tokens under predefined aspect ratios. A main layout branch coordinates the global document structure and dynamically forks multiple content branches for concurrent localized decoding, while reusing the shared-prefix KV cache. Within each branch, P-MTP further shortens the decoding path through progressive look-ahead prediction. In (b), P-MTP comprises a lightweight prediction module and a progressive loss-weighting strategy for supervising predictions at increasing look-ahead depths.

Decoding Roles and Sequence Formats. The hierarchical decoding process comprises two types of branches:

- **Layout branch.** The main layout branch generates a structured sequence following the natural reading order of the page. Each layout unit is represented by its category, normalized spatial coordinates, and a routing token `<FORK>`, which indicates that the corresponding regional content can be decoded by a separate branch.
- **Content branch.** Each local content branch generates the content associated with one layout unit. Its input consists of the shared visual context and the structural prefix corresponding to that unit, followed by a transition token `<CHILD>` that marks the beginning of localized content generation.

Branch-specific Sequence Supervision. Based on the two sequence formats, each original document annotation is transformed into a layout-branch sequence and a set of block-specific content-branch sequences. For layout parsing, all valid tokens in the structural sequence are supervised. For content decoding, the visual context and block-specific structural prefix serve only as conditioning information, while supervision is restricted to the localized transcription following `<CHILD>`. The corresponding validity mask is defined as

$$\mathcal{M}_t^{\text{dec}} = \mathbb{I}(t > t_{\langle \text{CHILD} \rangle}), \tag{1}$$

where $t_{\langle \text{CHILD} \rangle}$ denotes the position of `<CHILD>` in the content-decoding sequence. This masking strategy allows the structural prefix to condition localized generation without requiring the content branch to reproduce the shared context.

Dynamic Branch Forking and Efficient Context Isolation. During inference, HPD-Parsing maintains a main layout branch that progressively predicts document regions in reading order. Whenever the layout branch emits a <FORK> token, the inference scheduler creates a localized content branch for the corresponding region. The new branch reuses the available visual and structural-prefix KV cache, replaces <FORK> with <CHILD>, and begins generating the regional content. Meanwhile, the main layout branch continues to identify subsequent regions, allowing multiple content branches to decode concurrently.

This design avoids redundant prefix computation. Each content branch shares the visual context and the layout prefix available at its forking position, while maintaining only its own incremental content KV cache. Consequently, a newly created branch does not need to re-encode the document image or repeat the common prefill computation. More importantly, each content branch is isolated from the detailed textual histories of unrelated regions. Under conventional full-page autoregressive decoding, every newly generated token attends to the complete preceding output sequence, causing the active KV cache and per-step attention cost to grow continuously with document length. In HPD-Parsing, each branch attends only to the shared visual context, its associated structural prefix, and its own localized generation history. This combination of shared-prefix reuse and localized context isolation shortens the active attention horizon and enables efficient concurrent decoding.

3.4. Integration with P-MTP

Layout-coordinated parallel decoding removes unnecessary sequential dependencies across document regions, but token generation within each active branch remains autoregressive. To further shorten the intra-branch decoding trajectories, HPD-Parsing integrates Progressive Multi-Token Prediction (P-MTP) into both the main layout branch and localized content branches.

P-MTP employs a lightweight residual MLP to progressively predict multiple future tokens from the decoder hidden states. At decoding position t and look-ahead depth k, the module combines the preceding hidden representation with the embedding of the previously predicted token, and projects the resulting feature through the shared LM head to obtain the speculative distribution $\hat{p}_t^k$. To stabilize predictions at increasing look-ahead depths, P-MTP adopts progressive loss weighting that assigns larger weights to reliable speculative trajectories.

Under hierarchical parallel decoding, standard next-token prediction and progressive look-ahead prediction are jointly optimized across all decoding branches:

$$\mathcal{L} = \sum_{k=0}^{K} \frac{\sum_{t=0}^{N} \mathcal{M}_{t+k+1} \mathcal{W}_{t,k} \ell\left(\hat{p}_t^k, y_{t+k+1}\right)}{\sum_{t=0}^{N} \mathcal{M}_{t+k+1}}, \tag{2}$$

where $k = 0$ denotes the standard next-token objective, while $k \geqslant 1$ denotes progressive multi-token prediction. We set $\mathcal{W}_{t,0} = 1$ for standard next-token prediction. For $k \geqslant 1$, $\mathcal{W}_{t,k}$ denotes the progressive loss weight assigned to the depth-k prediction at position t, which combines a distance-dependent decay with path- and target-consistency signals to down-weight unreliable long-range predictions as detailed in our previous work [22]. The mask $\mathcal{M}_{t+k+1}$ specifies whether the target position is valid within the corresponding decoding branch. All valid positions are supervised for full-page and layout parsing, whereas content branches follow the content-validity rule defined in Equation 1.

During inference, P-MTP allows each active branch to draft multiple future tokens, verify them in parallel, and accept several tokens within a single decoding step. With an average accepted length of **6.6** tokens per step, it substantially reduces the decoding iterations required

within both layout and content branches. Collectively, layout-coordinated branch concurrency and P-MTP shorten the decoding path across branches and within each branch, respectively.

4. Model Training and Data Curation

HPD-Parsing is built on the open-source InternVL-3.5 1B and further adapted through a staged training strategy supported by a scalable data curation pipeline. This section first describes how the model transitions from conventional full-page parsing to hierarchical parallel decoding, and then introduces the construction and refinement of the corresponding training data.

4.1. Staged Adaptation to Hierarchical Parallel Decoding

The training strategy of HPD-Parsing is designed to progressively acquire general document parsing capability, adapt the model to hierarchical parallel decoding, and improve task-specific output quality. Conventional full-page sequence generation provides dense supervision over complete parsing outputs and is therefore suitable for learning broad recognition, structural understanding, and reading-order modeling. However, hierarchical parallel decoding introduces distinct decoding roles for global layout coordination and localized content generation, which require branch-specific supervision. Moreover, the explicit decomposition of document outputs makes component-level reward assignment more tractable. Based on these considerations, we adopt a three-stage training strategy.

- **Stage 1: Document Parsing Capacity Initialization.** In the first stage, HPD-Parsing is trained using the conventional full-page sequence generation format to acquire general document parsing capability. Each document image is paired with its complete parsing output, providing dense supervision for text recognition, layout understanding, reading-order modeling, and structured generation. This formulation allows the model to learn from complete page-level sequences before being exposed to the hierarchical decoding format. Both the backbone and the P-MTP module are optimized under the joint objective in Equation 2, where all valid positions in the full-page output are supervised. This stage also enables P-MTP to acquire progressive look-ahead prediction capability from large-scale sequential outputs.
- **Stage 2: Decoding Paradigm Shift and Hard-case Optimization.** Starting from the full-page parser initialized in Stage 1, the second stage adapts the model to hierarchical parallel decoding through branch-specific supervision. Each page is reorganized into two types of training instances corresponding to the decoding roles introduced in Section 3.3: global layout parsing and localized content decoding. Layout instances supervise the complete structural sequence, allowing the main branch to learn document organization, region assignment, and reading order. Content instances apply the validity mask in Equation 1, such that only the localized transcription following `<CHILD>` contributes to the objective. P-MTP is optimized under the same masks, extending each branch from next-token prediction to progressive look-ahead prediction. Since the model has already acquired general parsing capability, this stage primarily focuses on learning the new decoding format and strengthening performance on challenging document structures.
- **Stage 3: Reward-Guided Optimization.** In the final stage, we perform lightweight reinforcement learning to improve output consistency and failure-prone parsing behaviors. The hierarchical output structure allows reward signals to be assigned to different parsing components more directly. We therefore construct task-aware rewards for formula quality, table parsing, and layout prediction, together with count-based consistency rewards.

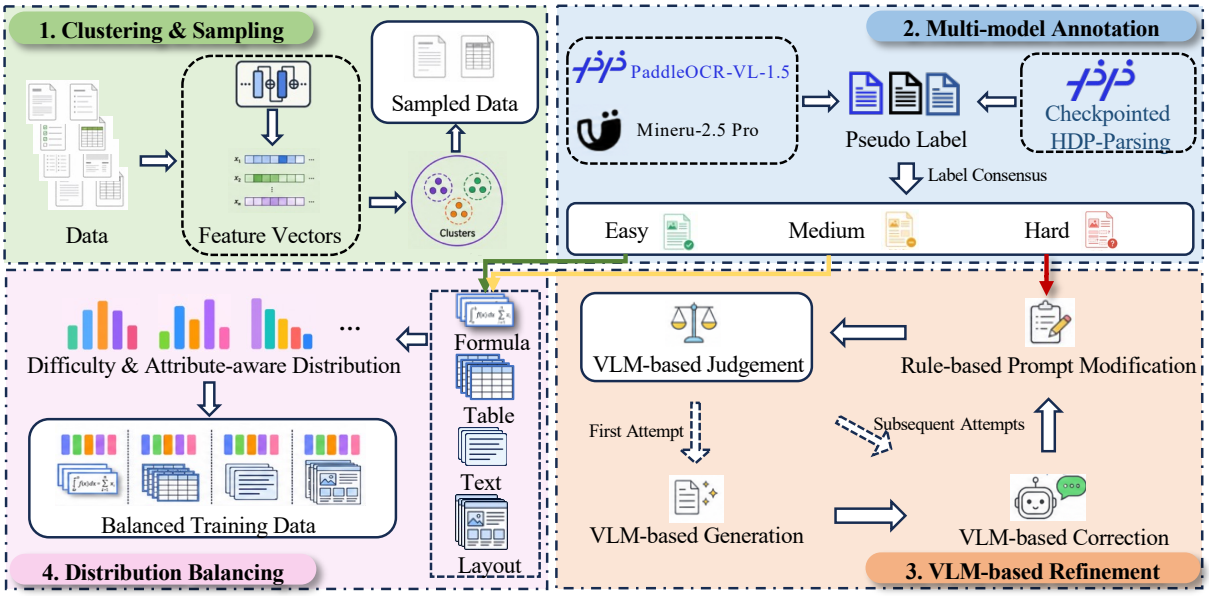


Figure 4 | Overview of the automated difficulty-aware data curation pipeline, which consists of four stages: Clustering and Sampling, Multi-model Annotation and Difficulty Estimation, VLM-based Refinement and Distribution Balancing.

These signals provide fine-grained feedback on recognition fidelity, structural accuracy, and output conformity, enabling targeted optimization without substantially altering the general parsing capability acquired in the preceding stages.

4.2. Automated Difficulty-Aware Data Curation Pipeline

To construct diverse and reliable training data for the different training stages of HPD-Parsing, we develop an automated difficulty-aware data curation pipeline, as illustrated in Figure 4. The pipeline consists of four stages: feature-based clustering and sampling, multi-model annotation and difficulty estimation, VLM-based refinement, and distribution balancing.

Clustering and Sampling. The first stage aims to improve data coverage while reducing redundancy. We extract visual feature representations from raw document images and group samples with similar document characteristics into clusters. Representative samples are then selected from different clusters, preventing the curated dataset from being dominated by highly repetitive layouts or visually similar pages. This process provides a diverse and scalable data pool for subsequent annotation.

Multi-model Annotation and Difficulty Estimation. The sampled documents are annotated using complementary open-source document parsing models, including PaddleOCR-VL-1.5 and MinerU-2.5 Pro. Their predictions are used to construct initial pseudo-labels. Meanwhile, an intermediate HPD-Parsing checkpoint generates parsing results for the same samples. We evaluate the checkpoint predictions against the pseudo-labels using the OmniDocBench evaluation protocol and categorize samples into Easy, Medium, and Hard groups according to their parsing discrepancies. Samples that can be reliably parsed by the intermediate model are assigned to the easier groups, whereas samples exhibiting textual or structural errors are treated as hard cases.

VLM-based Refinement. Hard samples are further processed through an iterative refinement

procedure. A stronger VLM first examines the current annotation and identifies potential errors in text recognition, structure, or formatting. Based on the detected error types, the prompt is adjusted using predefined rules and the annotation is regenerated or corrected. The refined result is then evaluated again by the VLM. This process continues until the annotation satisfies the quality criterion or reaches a maximum of N refinement attempts. The refinement stage is particularly beneficial for complex samples containing dense tables, multi-column layouts, formulas, or ambiguous region structures.

Distribution Balancing. After annotation and refinement, the curated samples are balanced across both difficulty levels and document attributes. We regulate the proportions of Easy, Medium, and Hard samples to prevent the training data from being dominated by straightforward cases. We further balance the occurrence of major document elements and layout characteristics, including text blocks, tables, formulas, and complex layouts. The resulting dataset provides broader structural coverage and more reliable supervision for training under the hierarchical parallel decoding paradigm.

The pipeline is instantiated differently across the three training stages described in Section 4.1. For Stage 1, we construct 2.8 million representative full-page training samples through large-scale feature clustering, filtering, and sampling, with annotations generated primarily by MinerU-2.5 Pro. This stage emphasizes broad document coverage and scalable capability initialization. For Stage 2, we construct 100K branch-specific samples from the more thoroughly curated portion of the data pool. These samples undergo multi-model annotation, difficulty estimation, distribution balancing, and, for challenging cases, iterative refinement with a stronger VLM before being converted into the global layout parsing and localized content decoding formats. For Stage 3, we further select 600 representative hard cases from the refined data pool and organize them according to component-specific error patterns, providing targeted samples for task-aware reward optimization.

5. Inference Workflow

To realize Hierarchical Parallel Decoding during inference, we extend the standard first-come-first-served (FCFS) scheduling framework with Layout-Coordinated Parallel Decoding and P-MTP. The resulting workflow dynamically forks concurrent content branches with shared-prefix KV cache reuse and accelerates token generation across all decoding branches, as formalized in Algorithm 1.

Algorithm 1: FCFS Scheduling + Hierarchical Parallel Decoding

Input: KV occupancy threshold τ, draft window K, concurrency cap $N_{\max}$
Output: Decoded output sequences for all requests
Data: $\mathcal{P}$: Active processing pool; $\mathcal{Q}$: Waiting queue; $\mathcal{M}[r]$: KV blocks of request r; *queue_dirty* $\in \{\texttt{true}, \texttt{false}\}$: Queue status flag; fork_rel$[p]$: Active children of parent p

while engine is running **do**
  `// FCFS Scheduling.`
  if queue_dirty **then**
    foreach $r \in \mathcal{Q}$ **do**
      if *r.is_child* **then**
        $p \leftarrow r.parent\_id, \quad m \leftarrow |\,\text{fork\_rel}[p]|;$
        $key(r) \leftarrow (0,\ arrival(p),\ -m,\ arrival(r));$
      else
        $key(r) \leftarrow (1,\ arrival(r));$
    $\mathcal{Q} \leftarrow \text{sort\_by\_key}(\mathcal{Q}); \quad queue\_dirty \leftarrow \texttt{false}\,;$ `// Sort waiting queue`
  foreach $r \in \mathcal{P}$ **do**
    while AllocKV$(\mathcal{M}[r]) = \texttt{null}$ **do**
      $v \leftarrow \arg\max_{r' \in \mathcal{P}} arrival(r')\,;$ `// Preempt youngest`
      PartialFree$(\mathcal{M}[v])$; $\mathcal{Q}.\text{prepend}(v)$;
      if $v = r$ **then break**;
  while $\mathcal{Q} \neq \emptyset$ **do**
    $r \leftarrow \mathcal{Q}.\text{peek}()$;
    if $\neg r.is\_child$ **and** (KVUsage(KVPool) $> \tau$ **or** $|\mathcal{P}| \geq N_{\max}$) **then break**;
    $\mathcal{Q}.\text{pop}()$; $\mathcal{P}.\text{append}(r)$; `// Gate parents only; children bypass`
  `// P-MTP based Speculative Decoding & Verification`
  parallel for $r \in \mathcal{P}$ **do** $r.\text{spec\_tokens} \leftarrow f_d(r.\text{context})$;
  foreach $r \in \mathcal{P}$ **do**
    $n_{\text{acc}} \leftarrow \max\{k \mid \boldsymbol{y}^*_{n+i} = \hat{t}_i,\ \forall i \leqslant k\}; \quad r.len \leftarrow r.len + n_{\text{acc}} + 1 - (K - n_{\text{acc}});$
    WriteKV$(\mathcal{M}[r],\ r.len)$;
  `// Fork Detection & KV Zero-Copy Sharing`
  foreach $r \in \mathcal{P},\ t \in \boldsymbol{Y}_r$ **do**
    if $t =$ FORKID **and** $\neg r.is\_child$ **then**
      $c \leftarrow r.\text{SpawnChild}(); \quad \text{KVManager.fork\_ptr}(\mathcal{M}[r], \mathcal{M}[c]);$
      $\mathcal{Q}.\text{add}(c); \quad \text{fork\_rel}[r.id].\text{add}(c); \quad queue\_dirty \leftarrow \texttt{true};$
  `// Reference-Counted Resource Release`
  foreach $r \in \mathcal{P}$ **with** $r.status =$ DONE **do**
    $\mathcal{P}.\text{remove}(r)$;
    foreach $b \in \mathcal{M}[r]$ **do**
      $b.ref\_count \leftarrow b.ref\_count - 1;$
      if $b.ref\_count = 0$ **then** KVPool.free(b);
    if $r.is\_child$ **then**
      $\text{fork\_rel}[r.parent\_id].\text{remove}(r)$;
    if $|\,\text{fork\_rel}[r.parent\_id]| = 0$ **then**
      Emit(InterleaveOutputs$(r.parent_id)$);
      CleanupForkData$(r.parent_id)$;

6. Experiments

6.1. Experimental Setup

Implementation Details. We conduct all experiments on 8 NVIDIA A800 GPUs, each with 80GB of memory. Our model is built upon OpenGVLab/InternVL3.5-1B, and all parameters are optimized without freezing the vision encoder or aligner. The training process consists of three stages. In the first stage, Document Parsing Capacity Initialization, the model acquires general document parsing capabilities with an initial learning rate of 1×10^{-4}. In the second stage, Decoding Paradigm Shift and Hard-case Optimization, the model is adapted to the proposed hierarchical parallel decoding paradigm and further optimized on challenging cases using an initial learning rate of 1×10^{-5}. In the third stage, Reward-Guided Optimization., we further refine the model through reinforcement learning with a learning rate of 5×10^{-7} and a global batch size of 96. We use bfloat16 precision throughout all three stages. For the first two stages, we train for 1 epoch, set the per-device batch size to 2, and use 8 gradient accumulation steps, resulting in an effective global batch size of 128. The maximum sequence length is set to 16,000 tokens, with a warmup ratio of 0.05. We adopt DeepSpeed ZeRO-1 with gradient checkpointing to reduce memory consumption and use FlashAttention for efficient attention computation.

For inference, we build our deployment system upon vLLM 0.17.1 and evaluate the throughout on NVIDIA A800 GPUs with 80GB memory. We implement the proposed hierarchical decoding mechanism within the vLLM serving framework, while preserving its first-come-first-served scheduling policy to maintain stable online serving behavior. The maximum generation length is set to 8,000 tokens for each decoding branch.

Evaluation Metrics. We evaluate our model on widely-adopted document parsing benchmarks: OmniDocBench, which comprehensively assess document parsing capabilities across multiple dimensions including text recognition, formula recognition, table structure extraction, and reading order prediction. Following standard practices in document parsing literature, we adopt task-specific metrics for comprehensive evaluation: (1) **Text Edit Distance** (Edit↓) measures character-level accuracy for text recognition; (2) **Formula CDM** (CDM↑) evaluates mathematical formula recognition quality; (3) **Table TEDS** (TEDS↑) and **Table TEDS-S** (TEDS-S↑) assess table structure extraction accuracy with and without content recognition; (4) **Reading Order Edit Distance** (Edit↓) quantifies the correctness of predicted reading sequences. The **Overall** score is computed as the weighted average across text, formula and table recognition.

6.2. Main Results

Effectiveness Comparison. Table 1 compares HPD-Parsing with state-of-the-art document parsing methods on OmniDocBench v1.6. By combining global layout coordination with localized parallel decoding, HPD-Parsing preserves page-level structure while reducing the impact of difficult regions on the overall output. With only 1B parameters, it achieves an overall score of 94.91, establishing a new state of the art among end-to-end unified parsers and outperforming larger models such as Qianfan-OCR, Logics-Parsing-v2, and FireRed-OCR. It also remains competitive with leading pipeline-based parsers despite using substantially fewer training samples, achieving a ReadOrderEdit score of 0.124. This suggests that decomposing layout and content supervision enables more targeted optimization, while the unified architecture may better preserve global document context by allowing each region to be interpreted with awareness of the entire page.

Efficiency Comparison. Table 2 compares the inference efficiency of different document parsing

Table 1 | Performance comparison on OmniDocBench v1.6. Best results in each category are in **bold**.

Model Type	Methods	Size	Overall↑	TextEdit↓	FormulaCDM↑	TableTEDS↑	TableTEDS-S↑	ReadOrderEdit↓
General VLMs	InternVL3.5-241B [24]	241B	83.76	0.130	89.95	74.35	79.78	0.215
	Kimi K2.5 [25]	1T	84.53	0.107	83.50	80.76	84.00	0.211
	GPT-5.2 [26]	-	86.59	0.114	88.21	82.95	87.93	0.193
	Qwen3-VL-235B [27]	235B	89.78	0.063	92.55	83.07	86.75	0.166
	Gemini 3 Flash [28]	-	92.62	0.066	95.16	89.29	93.51	0.172
	Gemini 3 Pro [28]	-	92.91	0.064	95.99	89.15	92.96	0.165
	Ovis2.6-30B-A3B [29]	30B	93.70	0.035	95.17	89.44	92.40	0.135
Specialized VLMs (Pipeline)	Dolphin-1.5 [7]	0.3B	86.52	0.094	87.49	81.43	84.82	0.167
	MonkeyOCR-pro-3B [30]	3B	88.57	0.074	88.74	84.35	88.62	0.189
	Dolphin-v2 [8]	3B	89.50	0.069	91.01	84.40	87.44	0.150
	OpenDoc-0.1B [31]	0.1B	90.67	0.049	93.02	83.88	87.45	0.140
	MinerU-2.5 [32]	1.2B	93.04	0.045	95.77	87.88	91.47	0.130
	Youtu-Parsing [6]	2.5B	93.74	0.044	93.63	92.02	95.00	0.116
	PaddleOCR-VL [1]	0.9B	94.18	0.040	95.91	90.65	93.74	0.135
	PaddleOCR-VL-1.5 [2]	0.9B	94.93	0.038	96.89	91.67	94.37	0.130
	GLM-OCR [9]	0.9B	95.22	0.044	97.18	92.83	95.39	0.133
	MinerU2.5-Pro [4]	1.2B	95.75	0.036	97.45	93.42	95.92	0.120
	PaddleOCR-VL-1.6 [33]	0.9B	96.3	0.032	97.5	94.8	97.11	0.127
Specialized VLMs (Unified)	POINTS-Reader [34]	3B	83.37	0.096	85.72	73.98	77.40	0.198
	Nanonets-OCR-s [35]	3B	83.61	0.108	81.46	80.18	84.51	0.213
	olmOCR [36]	7B	85.74	0.139	88.10	83.00	87.17	0.216
	OCRVerse [37]	4B	88.60	0.063	89.61	82.44	86.27	0.163
	DeepSeek-OCR 2 [18]	3B-A0.5B	90.25	0.050	91.84	83.89	87.75	0.144
	dots.ocr [13]	3B	90.77	0.048	89.95	87.18	90.58	0.138
	FireRed-OCR [10]	2B	93.26	**0.037**	95.44	88.04	91.06	0.131
	Logics-Parsing-v2 [19]	4B	93.33	0.041	95.65	88.42	91.98	0.137
	Qianfan-OCR [11]	4B	93.90	0.040	95.08	90.53	93.31	0.130
	Unlimited OCR [20]	3B-A0.5B	93.92	0.042	95.79	90.16	93.32	0.129
	HunyuanOCR-1.5 [21]	1B	94.74	0.039	94.50	**93.67**	**94.71**	0.129
	Ours (HPD-Parsing)	1B	**94.91**	0.039	**97.28**	91.35	94.11	**0.124**

methods on OmniDocBench v1.6, together with their average input token counts. Compared to the autoregressive baseline, HPD-Parsing increases throughput from 1.02 to 2.68 PPS and from 1,554.8 to 4,752.1 TPS, corresponding to improvements 2.62× and 3.06×, respectively. This substantial gain is achieved by replacing the conventional single autoregressive trajectory with hierarchical parallel decoding, which enables concurrent decoding across layout-coordinated content branches and further reduces the decoding steps within each branch through P-MTP. Input token count provides additional context for interpreting the efficiency comparison, as longer inputs generally incur greater prefilling and attention overhead. Despite processing approximately 4,800 input tokens per page, over four times that of DeepSeek-OCR-2, HPD-Parsing still achieves 1.31× higher PPS and 1.62× higher TPS, demonstrating strong inference efficiency under a considerably larger input-token budget.

To examine how the acceleration gains vary with generation length, we partition the OmniDocBench v1.6 test set into output-length buckets and compare HPD-Parsing with the vanilla autoregressive baseline. As shown in Figure 5, the efficiency advantage becomes increasingly pronounced as the output sequence grows. Conventional unified document parsers generate the entire page along a single autoregressive trajectory, causing the number of decoding steps to increase approximately with the total output length. In contrast, HPD-Parsing uses the main layout branch for global coordination and delegates localized content decoding to concurrent branches. Consequently, the critical decoding path is primarily determined by the longest active branch rather than the sum of all block lengths. P-MTP further shortens each local decoding trajectory by generating and accepting multiple tokens at each step. By combining layout-coordinated branch concurrency with progressive multi-token prediction, HPD-Parsing alleviates both page-level sequential execution and token-by-token decoding within individual branches. Its acceleration advantage therefore grows with document length, reaching up to 18.04× fewer decoding steps, 3.67× higher request throughput, and 5.80× lower single-request latency in the longest output-length bucket. These results demonstrate that hierarchical paral-

Table 2 | Inference speed comparison on OmniDocBench v1.6. TPS(tokens per second), PPS(pages per second) under batch size 512.

Model Type	Method	Size	Avg Input Tokens	TPS↑ (BS=512)	PPS↑ (BS=512)
Specialized VLMs (Pipeline)	Youtu-Parsing [6]	2.5B	-	315.4	0.39
	MinerU2.5-Pro [4]	1.2B	-	1890.3	1.58
	PaddleOCR-VL-1.6 [33]	0.9B	-	1533.8	1.25
	GLM-OCR [9]	0.9B	-	2133.8	1.86
Specialized VLMs (Unified)	Qianfan-OCR [11]	4B	1856.9	994.47	0.60
	FireRed-OCR [10]	2B	4532.3	1082.0	0.90
	HunyuanOCR-1.5 [21]	1B	4496.9	1081.3	1.10
	DeepSeek-OCR-2 [18]	3B-A0.5B	1100.2	2932.1	2.05
	Unlimited OCR [20]	3B-A0.5B	1485.6	2901.5	2.03
	Baseline (Autoregressive)	1B	4809.3	1554.8	1.02
	Ours (HPD-Parsing)	1B	4809.3	**4752.1**	**2.68**

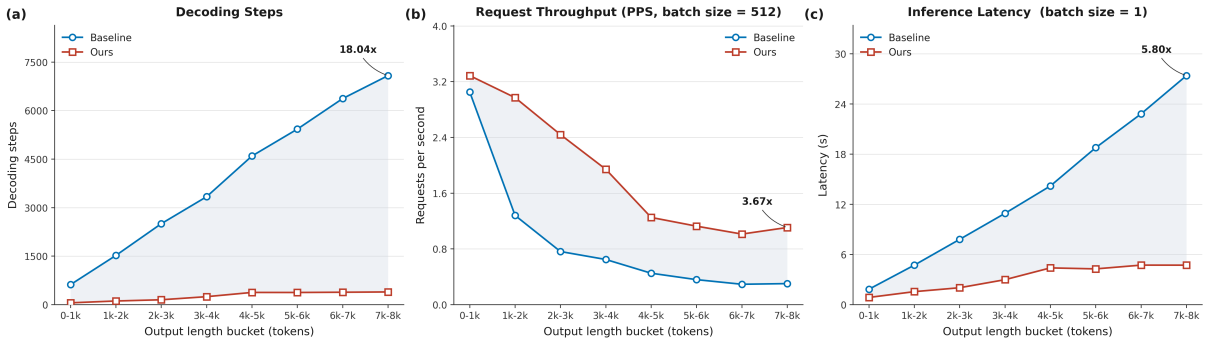


Figure 5 | **Efficiency scaling across output-length buckets.** Compared with the autoregressive baseline, HPD-Parsing exhibits increasingly larger efficiency gains as output length grows. It reduces decoding steps by up to 18.04× in (a), improves request throughput by up to 3.67× under batch size 512 in (b), and reduces single-request inference latency by up to 5.80× under batch size 1 in (c).

lel decoding effectively mitigates the sequence-length-dependent slowdown of conventional autoregressive document parsing.

6.3. Qualitative Analysis

Comprehensive Coverage of Complex Documents. Our method demonstrates robust parsing performance across document pages with diverse structural characteristics. Figure 6 presents representative results on documents with complex mathematical expressions. The model accurately transcribes long multi-line formulas, matrix equations, fractions, and densely arranged mathematical symbols, while preserving valid LaTeX structures for faithful rendering. Figure 7 further evaluates the model on table-intensive documents. The results show that it can recover large and densely populated tables, including multi-row and multi-column structures, in a structured HTML representation while maintaining the original cell organization. Finally, Figure 8 illustrates performance on pages with challenging layouts including newspapers and puzzle-style documents. The model preserves the spatial arrangement and reading order of interleaved text blocks, figures, and other visual elements, enabling coherent reconstruction of the overall page structure. Together, these qualitative examples show that the unified parsing framework can consistently process heterogeneous document content within full-page transcription.

Comparative Advantages over Competing Methods. We provide qualitative comparisons to

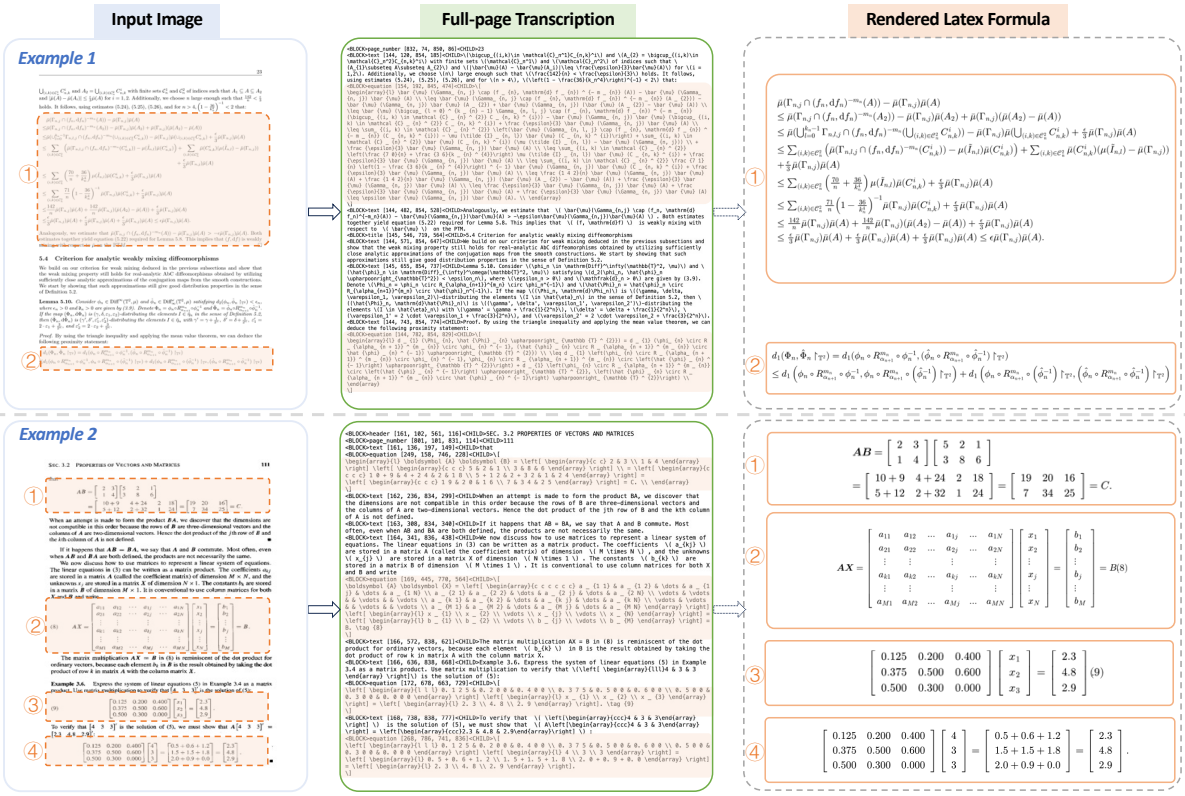


Figure 6 | Qualitative results on document images with complex formulas.

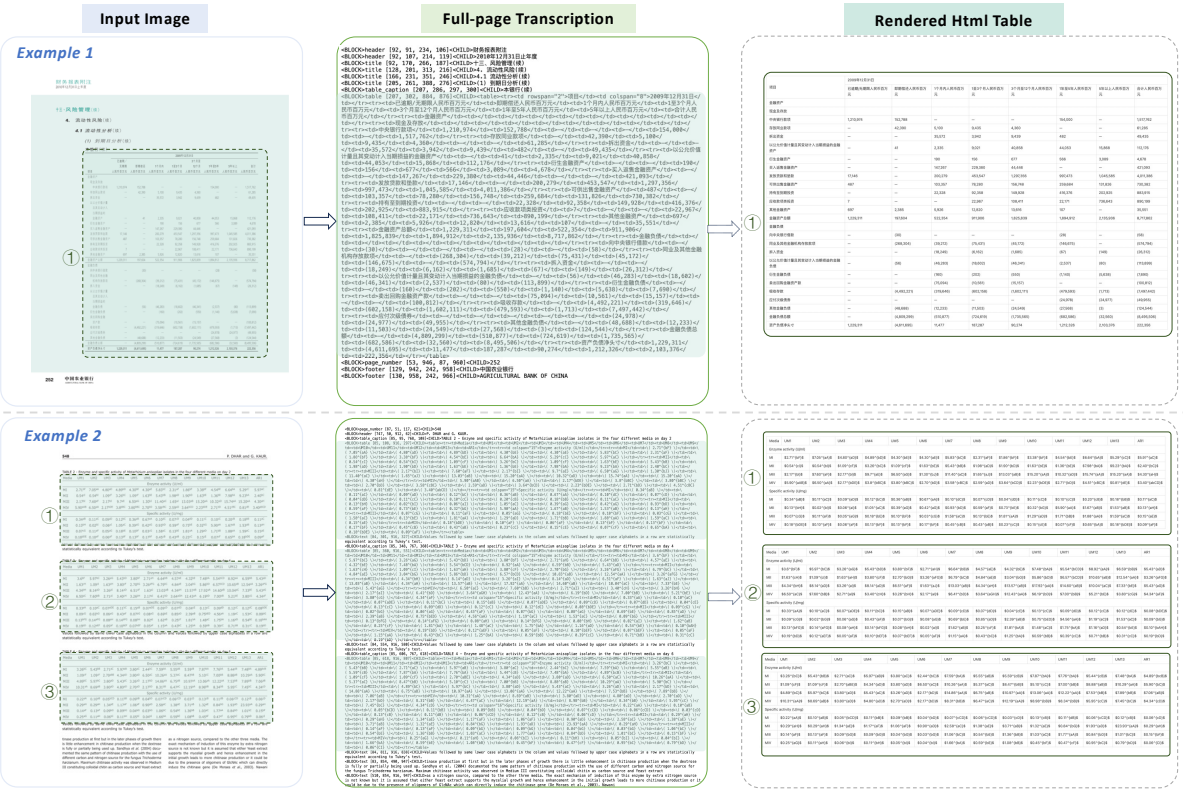


Figure 7 | Qualitative results on document images with complex tables.

illustrate the key advantages of our hierarchical parallel decoding paradigm over both traditional pipeline-based parsers and competing unified VLM approaches. Figure 9 presents representative failure cases of competing methods alongside our successful parsing results, highlighting three critical benefits:

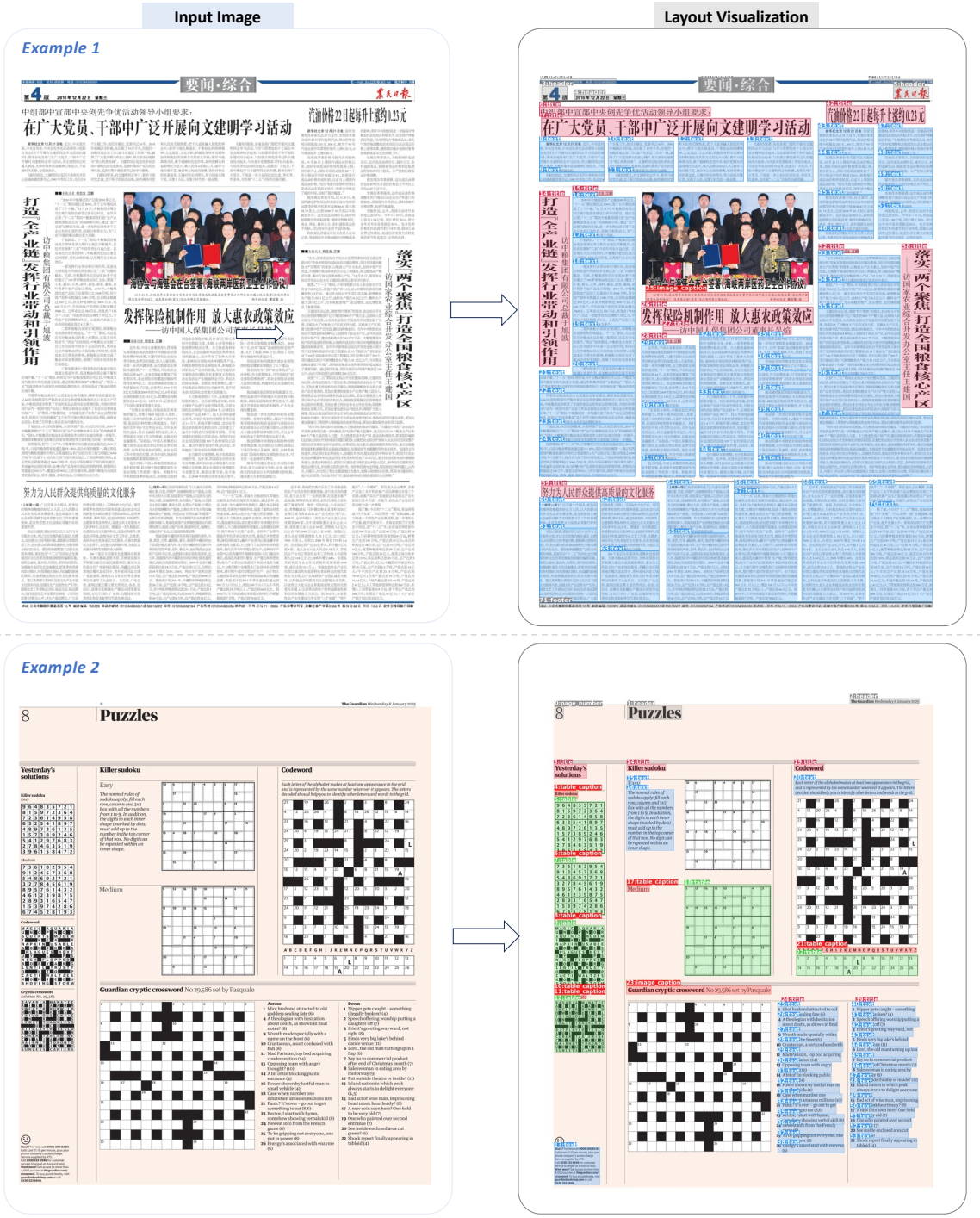


Figure 8 | Qualitative results on document images with complex layout.

- **Robustness to Detection Errors.** Pipeline-based methods depend heavily on the layout analysis stage. As shown in Figure 9(a), inaccurate or incomplete bounding boxes may

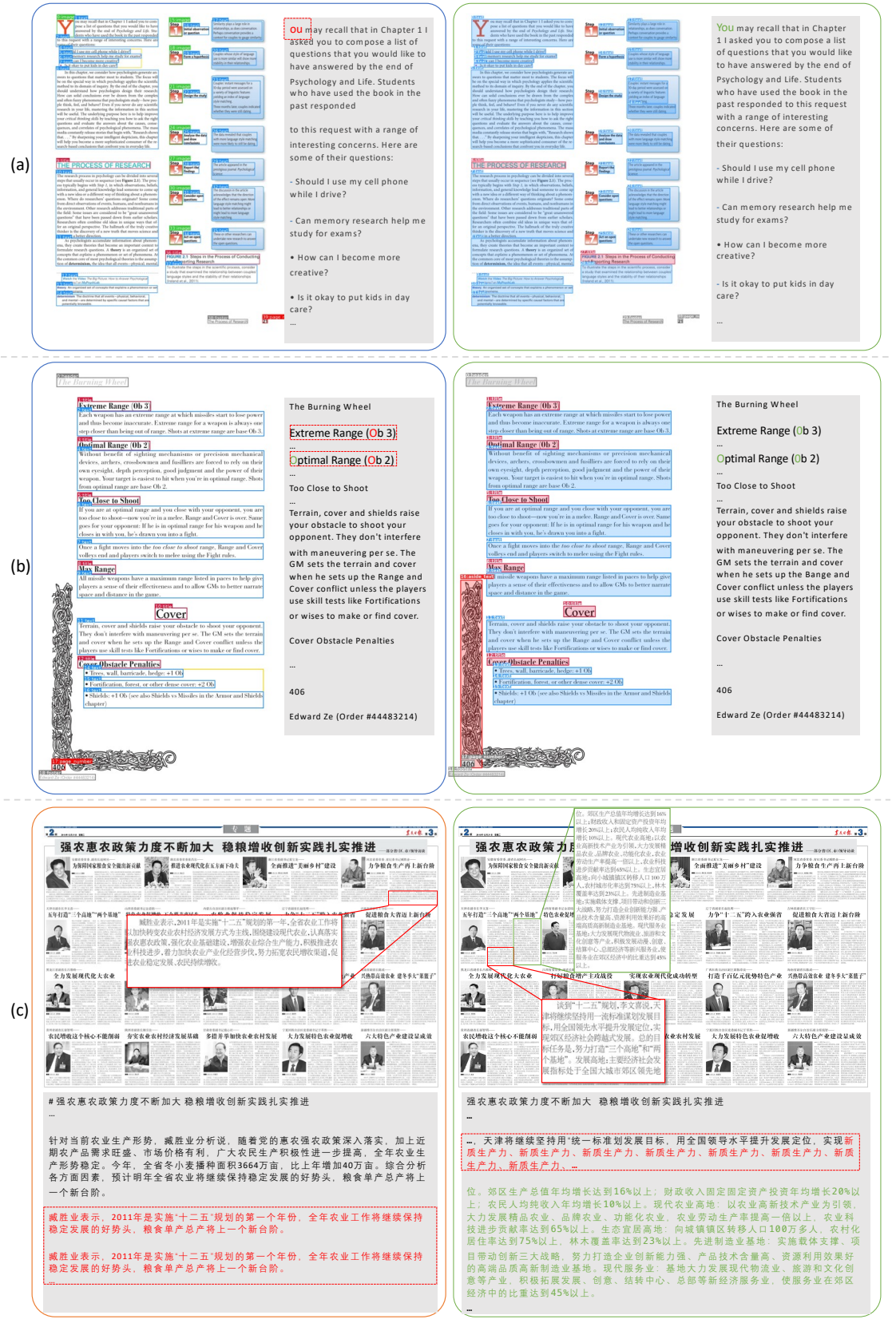


Figure 9 | Comparative advantages over competing methods. HPD-Parsing demonstrates superior robustness in (a) handling detection errors caused by misaligned bounding boxes, (b) recognizing visually similar characters through global contextual reasoning, and (c) preventing repetition errors from propagating across document blocks.

exclude relevant content or mix adjacent regions, directly causing recognition errors in the subsequent stage. In contrast, HPD-Parsing incorporates document-level visual and semantic context during parsing, enabling correct recognition despite imperfect region localization.

- **Handling Visually Similar Characters.** Figure 9(b) presents a representative case involving the visually similar characters '0' and 'O'. Pipeline-based methods process isolated text regions with limited contextual information and may therefore confuse the digit zero with the uppercase letter O. By leveraging broader document context, HPD-Parsing more reliably disambiguates the two characters and produces the correct parsing result.
- **Error Propagation Prevention.** Unified autoregressive methods may enter repetitive decoding once an error occurs, causing the corrupted output to propagate through subsequent document content. As illustrated in Figure 9(c), competing methods repeatedly generate the same text span. In HPD-Parsing, localized content is decoded in independent branches, confining repetition errors to individual regions and preventing them from affecting subsequent document blocks.

7. Conclusion and Future Work

In this paper, we present HPD-Parsing, a lightweight and high-throughput document parser built upon a hierarchical parallel decoding paradigm. Motivated by the observation that document parsing requires global coordination while content decoding is largely localized, HPD-Parsing replaces the traditional single autoregressive trajectory with a main layout branch for global coordination and concurrent local decoding branches for region-level content generation. Progressive Multi-Token Prediction further reduces the decoding steps within both the layout and content branches. Experiments on public benchmarks demonstrate that HPD-Parsing substantially improves inference throughput while maintaining competitive parsing accuracy, validating layout-coordinated parallel concurrency as an effective alternative to fully sequential document generation. More generally, the globally coordinated yet locally decomposable structure of documents indicates that hierarchical parallel decoding may extend beyond single-page parsing to key information extraction, multi-page document understanding, and other structured generation tasks. In the future, we will further expand the training data to improve coverage across a wider range of document scenarios and explore structure-aware attention designs that reduce the effective attention context and the computation required at each decoding step.

References

[1] Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, et al. Paddleocr-vl: Boosting multilingual document parsing via a 0.9 b ultra-compact vision-language model. arXiv preprint arXiv:2510.14528, 2025.

[2] Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, et al. Paddleocr-vl-1.5: Towards a multi-task 0.9 b vlm for robust in-the-wild document parsing. arXiv preprint arXiv:2601.21957, 2026.

[3] Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, et al. Mineru: An open-source solution for precise document content extraction. arXiv preprint arXiv:2409.18839, 2024.

[4] Bin Wang, Tianyao He, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Tao Chu, Yuan Qu, Zhenjiang Jin, Weijun Zeng, Ziyang Miao, et al. Mineru2. 5-pro: Pushing the limits of data-centric document parsing at scale. arXiv preprint arXiv:2604.04771, 2026.

[5] Zhang Li, Yuliang Liu, Qiang Liu, Zhiyin Ma, Ziyang Zhang, Shuo Zhang, Biao Yang, Zidun Guo, Jiarui Zhang, Xinyu Wang, et al. Monkeyocr: Document parsing with a structure-recognition-relation triplet paradigm. arXiv preprint arXiv:2506.05218, 2025.

[6] Kun Yin, Yunfei Wu, Bing Liu, Zhongpeng Cai, Xiaotian Li, Huang Chen, Xin Li, Haoyu Cao, Yinsong Liu, Deqiang Jiang, et al. Youtu-parsing: Perception, structuring and recognition via high-parallelism decoding. arXiv preprint arXiv:2601.20430, 2026.

[7] Hao Feng, Shu Wei, Xiang Fei, Wei Shi, Yingdong Han, Lei Liao, Jinghui Lu, Binghong Wu, Qi Liu, Chunhui Lin, et al. Dolphin: Document image parsing via heterogeneous anchor prompting. In Findings of the Association for Computational Linguistics: ACL 2025, pages 21919–21936, 2025.

[8] Hao Feng, Wei Shi, Ke Zhang, Xiang Fei, Lei Liao, Dingkang Yang, Yongkun Du, Xuecheng Wu, Jingqun Tang, Yang Liu, et al. Dolphin-v2: Universal document parsing via scalable anchor prompting. arXiv preprint arXiv:2602.05384, 2026.

[9] Shuaiqi Duan, Yadong Xue, Weihan Wang, Zhe Su, Huan Liu, Sheng Yang, Guobing Gan, Guo Wang, Zihan Wang, Shengdong Yan, et al. Glm-ocr technical report. arXiv preprint arXiv:2603.10910, 2026.

[10] Hao Wu, Haoran Lou, Xinyue Li, Zuodong Zhong, Zhaojun Sun, Phellon Chen, Xuanhe Zhou, Kai Zuo, Yibo Chen, Xu Tang, et al. Firered-ocr technical report. arXiv preprint arXiv:2603.01840, 2026.

[11] Daxiang Dong, Mingming Zheng, Dong Xu, Chunhua Luo, Bairong Zhuang, Yuxuan Li, Ruoyun He, Haoran Wang, Wenyu Zhang, Wenbo Wang, et al. Qianfan-ocr: A unified end-to-end model for document intelligence. arXiv preprint arXiv:2603.13398, 2026.

[12] Hunyuan Vision Team, Pengyuan Lyu, Xingyu Wan, Gengluo Li, Shangpin Peng, Weinong Wang, Liang Wu, Huawen Shen, Yu Zhou, Canhui Tang, et al. Hunyuanocr technical report. arXiv preprint arXiv:2511.19575, 2025.

[13] Yumeng Li, Guang Yang, Hao Liu, Bowen Wang, and Colin Zhang. dots. ocr: Multilingual document layout parsing in a single vision-language model. arXiv preprint arXiv:2512.02498, 2025.

[14] Song Chen, Xinyu Guo, Yadong Li, Tao Zhang, Mingan Lin, Dongdong Kuang, Youwei Zhang, Lingfeng Ming, Fengyu Zhang, Yuran Wang, et al. Ocean-ocr: Towards general ocr application via a vision-language model. arXiv preprint arXiv:2501.15558, 2025.

[15] Lukas Blecher, Guillem Cucurull Preixens, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents. In International Conference on Learning Representations, volume 2024, pages 37646–37663, 2024.

[16] Said Taghadouini, Adrien Cavaillès, and Baptiste Aubertin. Lightonocr: A 1b end-to-end multilingual vision-language model for state-of-the-art ocr. arXiv preprint arXiv:2601.14251, 2026.

[17] Haoran Wei, Yaofeng Sun, and Yukun Li. Deepseek-ocr: Contexts optical compression. arXiv preprint arXiv:2510.18234, 2025.

[18] Haoran Wei, Yaofeng Sun, and Yukun Li. Deepseek-ocr 2: Visual causal flow. arXiv preprint arXiv:2601.20552, 2026.

[19] Xiangyang Chen, Shuzhao Li, Xiuwen Zhu, Yongfan Chen, Fan Yang, Cheng Fang, Lin Qu, Xiaoxiao Xu, Hu Wei, and Minggang Wu. Logics-parsing technical report. arXiv preprint arXiv:2509.19760, 2025.

[20] Youyang Yin, Huanhuan Liu, Qunyi Xie, Chaorun Liu, Shiqi Yang, Shaohua Wang, Zhanlong Liu, Hao Zou, Jinyue Chen, Shu Wei, et al. Unlimited ocr works. arXiv preprint arXiv:2606.23050, 2026.

[21] Gengluo Li, Xingyu Wan, Shangpin Peng, Weinong Wang, Hao Feng, Yongkun Du, Binghong Wu, Zheng Ruan, Zhiqiong Lu, Liang Wu, et al. Hunyuanocr-1.5: Making lightweight ocr vlms faster and better. arXiv preprint arXiv:2607.04884, 2026.

[22] Le Xiang, Chenxi Zhai, Shu Wei, Jingjing Wu, Qunyi Xie, Xiao Tan, Kunbin Chen, and Wei He. P-mtp: Efficient document parsing via multi-token prediction with progressive depth scaling. arXiv preprint arXiv:2606.24447, 2026.

[23] Pengyuan Lyu, Yulin Li, Hao Zhou, Weihong Ma, Xingyu Wan, Qunyi Xie, Liang Wu, Chengquan Zhang, Kun Yao, Errui Ding, et al. Structextv3: An efficient vision-language model for text-rich image perception, comprehension, and beyond. arXiv preprint arXiv:2405.21013, 2024.

[24] Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. arXiv preprint arXiv:2508.18265, 2025.

[25] Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2. 5: Visual agentic intelligence. arXiv preprint arXiv:2602.02276, 2026.

[26] Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. arXiv preprint arXiv:2601.03267, 2025.

[27] 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, 2025.

[28] Google. Gemini 3 Pro Model Card. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf, 2026.

[29] Shiyin Lu, Yang Li, Yu Xia, Yuwei Hu, Shanshan Zhao, Yanqing Ma, Zhichao Wei, Yinglun Li, Lunhao Duan, Jianshan Zhao, Yuxuan Han, Haijun Li, et al. Ovis2.5 Technical Report. arXiv:2508.11737, 2025.

[30] Zhang Li, Yuliang Liu, Qiang Liu, Zhiyin Ma, Ziyang Zhang, Shuo Zhang, Zidun Guo, Jiarui Zhang, Xinyu Wang, and Xiang Bai. MonkeyOCR: Document parsing with a structure-recognition-relation triplet paradigm. arXiv preprint arXiv:2506.05218, 2025.

[31] Yongkun Du, Zhineng Chen, Yazhen Xie, Weikang Bai, Hao Feng, Wei Shi, Yuchen Su, Can Huang, and Yu-Gang Jiang. Unirec-0.1b: Unified text and formula recognition with 0.1b parameters. arXiv preprint arXiv:2512.21095, 2025.

[32] Junbo Niu, Zheng Liu, Zhuangcheng Gu, Bin Wang, Linke Ouyang, Zhiyuan Zhao, Tao Chu, Tianyao He, Fan Wu, Qintong Zhang, et al. MinerU2.5: A decoupled vision-language model for efficient high-resolution document parsing. In Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics–Industry Track, 2025.

[33] Zelun Zhang, Hongen Liu, Suyin Liang, Yubo Zhang, Yiqing Xiang, Jiaxuan Liu, Ting Sun, Manhui Lin, Yue Zhang, Changda Zhou, et al. Paddleocr-vl-1.6: Expanding the frontier of document parsing with under-optimized region refinement and progressive post-training. arXiv preprint arXiv:2606.03264, 2026.

[34] Yuan Liu, Zhongyin Zhao, Le Tian, Haicheng Wang, Xubing Ye, Yangxiu You, Zilin Yu, Chuhan Wu, Zhou Xiao, Yang Yu, et al. POINTS-Reader: Distillation-free adaptation of vision-language models for document conversion. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2025.

[35] Souvik Mandal, Ashish Talewar, Paras Ahuja, and Prathamesh Juvatkar. Nanonets-OCR-S: A model for transforming documents into structured markdown with intelligent content recognition and semantic tagging, 2025. URL https://huggingface.co/nanonets/Nanonets-OCR-s.

[36] Jake Poznanski, Aman Rangapur, Jon Borchardt, Jason Dunkelberger, Regan Huff, Daniel Lin, Christopher Wilhelm, Kyle Lo, and Luca Soldaini. olmocr: Unlocking trillions of tokens in pdfs with vision language models. arXiv preprint arXiv:2502.18443, 2025.

[37] Yufeng Zhong, Lei Chen, Xuanle Zhao, Wenkang Han, Liming Zheng, Jing Huang, Deyang Jiang, Yilin Cao, Lin Ma, and Zhixiong Zeng. OCRVerse: Towards holistic OCR in end-to-end vision-language models. arXiv preprint arXiv:2601.21639, 2026.