FinixDoc: Rethinking Financial Document Parsing Beyond Saturated Benchmarks

Hang Wang[*], **Jin Zhang**[*], **Guoliang Xu**[*], **Pengyue Lu**
Yao Li, Zijiao Zhang, Tianyu Huang, Weiqi Xiong, Yulong Wang, Chuqiao Lu
Wenkang Huang, Kai Yang, Yadong Li, Hui Li, Xingzhong Xu[‡], **Xiao Xu**[✉]
[*]Equal contribution. [‡]Team leader. [✉]Corresponding author.
Ant Group, Hangzhou, China
✉ suyan.xx@antgroup.com

https://finix.alipay.com/
https://huggingface.co/datasets/inclusionAI/FinixDocBench

Abstract

Financial document parsing requires accuracy, structural consistency, and verifiability that current benchmarks often fail to reflect. We present **FinixDoc**, an end-to-end agentic parsing system for real-world financial documents, with **FinixDoc-VL**, a 4B-scale vision-language model built on Qwen3-VL-4B, as its core parser. To characterize the gap between benchmark and deployment performance, we introduce a **Document Parsing Capability Matrix** organized along two practical axes: visual quality and document scale. Guided by this matrix, FinixDoc-VL is trained with a domain-adapted recipe combining homoglyph-aware contrastive learning and multi-stage reinforcement learning with composite domain-specific rewards. To better leverage our accumulated advantage in low-quality financial-document data and support large-scale, high-quality data production, we further build a human-in-the-loop **Data Factory** pipeline with confidence-aware expert review. For evaluation, we construct **FinixDocBench**, a financial-domain evaluation suite covering digital-native, camera-captured, ultra-large-page, and internal-workflow scenarios, with a compliance-reviewed subset released alongside this technical report. On its main subsets, FinixDoc-VL achieves the highest overall score (**81.43**) among evaluated baselines, outperforming the next-best open-source model by 5.13 points, with the largest gains on internal financial workflows (**FinixInner: 84.08** vs. 78.73).

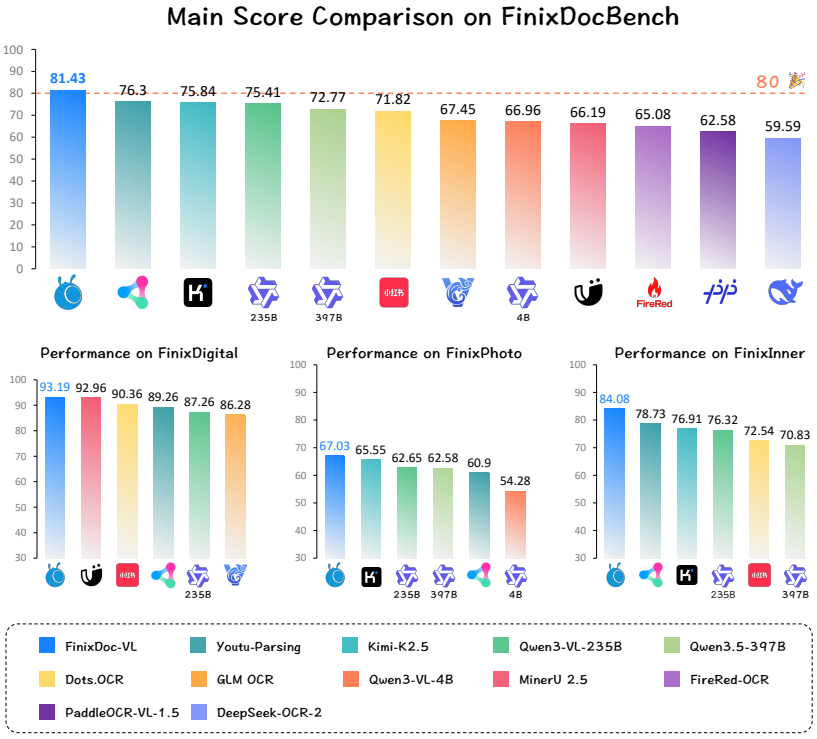

Figure 1: **Main score comparison on FinixDocBench.** The main score is computed as the macro-average over FinixDigital, FinixPhoto, and FinixInner. The top panel reports the overall main score, and the bottom panels show subset-level performance. FinixHuge is excluded from this comparison and reported separately, since it evaluates ultra-large-page documents under a system-level protocol with limited model coverage.

1 Introduction

As a financial technology enterprise operating large-scale financial workflows, we process massive volumes of heterogeneous financial documents every day, including insurance policies, medical receipts, claim materials, identity documents, expense statements, and financial reports. These documents arrive in diverse forms, such as digitally native PDFs, scanned copies, office files, and camera-captured images, and their parsed contents directly support downstream business processes that require high accuracy, structural consistency, traceability, and verifiability. This real-world industrial setting motivates our focus on financial document parsing.

Document parsing is therefore a foundational capability for document digitization, content understanding, and information extraction. Traditional approaches typically rely on task-specific pipelines composed of optical character recognition (OCR), layout analysis, and field extraction [32, 41, 4]. Although these methods can perform well in scenarios with well-defined rules, they usually face limited performance ceilings, high maintenance costs, and weak generalization across templates and application settings.

In recent years, Vision-Language Models (VLMs) have driven a major paradigm shift in document understanding [11, 14, 2, 1]. Unlike traditional pipelines that decouple recognition, structural analysis, and field extraction, modern document parsing is increasingly treated as a unified modeling problem [41, 11, 3, 18]. In this work, we use the term *parsing* to refer to this unified capability, which jointly encompasses text recognition, layout detection and categorization, table structure recovery, and reading-order reconstruction, with the final output rendered as either structured JSON or full-page Markdown.

In large-scale financial workflows, document parsing is not merely a generic OCR or layout-analysis task. Its outputs directly affect business processes and therefore impose stricter requirements than standard benchmark settings. These requirements expose failure modes that are often underrepresented in standard document benchmarks.

In real-world financial deployments, we observe a clear gap between benchmark performance and practical utility. Many models that achieve strong results on standard benchmarks (e.g., OmniDocBench [21]) still exhibit major shortcomings in production settings, including weak robustness, structural inconsistencies, and failures on long documents or large images. Such behavior falls well short of the stringent financial requirements for accuracy, consistency, and verifiability.

Through extensive analysis of failure cases in real financial deployments, we observe that these shortcomings are not isolated incidents but instead vary systematically along two orthogonal dimensions: the visual quality of the input document and the scale of the page or document. Accordingly, we introduce a *Document Parsing Capability Matrix* from the perspective of financial document processing. As illustrated in Figure 2, this matrix organizes document parsing scenarios along two dimensions: document quality and document scale.

Based on this matrix, we organize current document parsing scenarios into several representative capability zones. First, mainstream benchmarks are concentrated largely in the bottom-left region, which we term the **Benchmark-Converged Zone**. Documents in this region typically have clear page quality, regular layouts, sufficiently high resolution, and manageable scale. As a result, many VLMs, especially document-specific models, already achieve strong performance in this zone, and the differences among leading models are often small.

Second, along the horizontal axis, we define the **Low-Quality Zone**, which corresponds to scenarios with degraded document quality. This zone includes documents commonly encountered in real-world financial workflows, such as medical receipts, claim materials, identity cards, and reimbursement vouchers. Because these documents are often captured with mobile phones, they are frequently affected by blur, shadows, glare, creases, perspective distortion, and partial occlusion. In this zone, we observe substantial performance degradation across most existing solutions. More importantly, document-specific VLMs that perform strongly in the Benchmark-Converged Zone often underperform strong general-purpose VLMs under these conditions. We hypothesize that, compared with general-purpose VLMs, specialized document models typically have smaller parameter scales, narrower training objectives, and more limited data diversity, making them more sensitive to distribution shifts and less robust in challenging real-world conditions.

Third, in the upper region of the matrix, we define the **Underexplored Large-Scale Zone**, which corresponds to large-scale documents that remain insufficiently covered by existing benchmarks and systems. Representative examples include ultra-long insurance policies and densely structured multi-column tables. When a document becomes excessively large (e.g., over 100 million pixels), directly feeding the entire page into a single VLM poses a major challenge for most existing models. This difficulty arises either because aggressive downsampling removes critical local details, or because the resulting input and

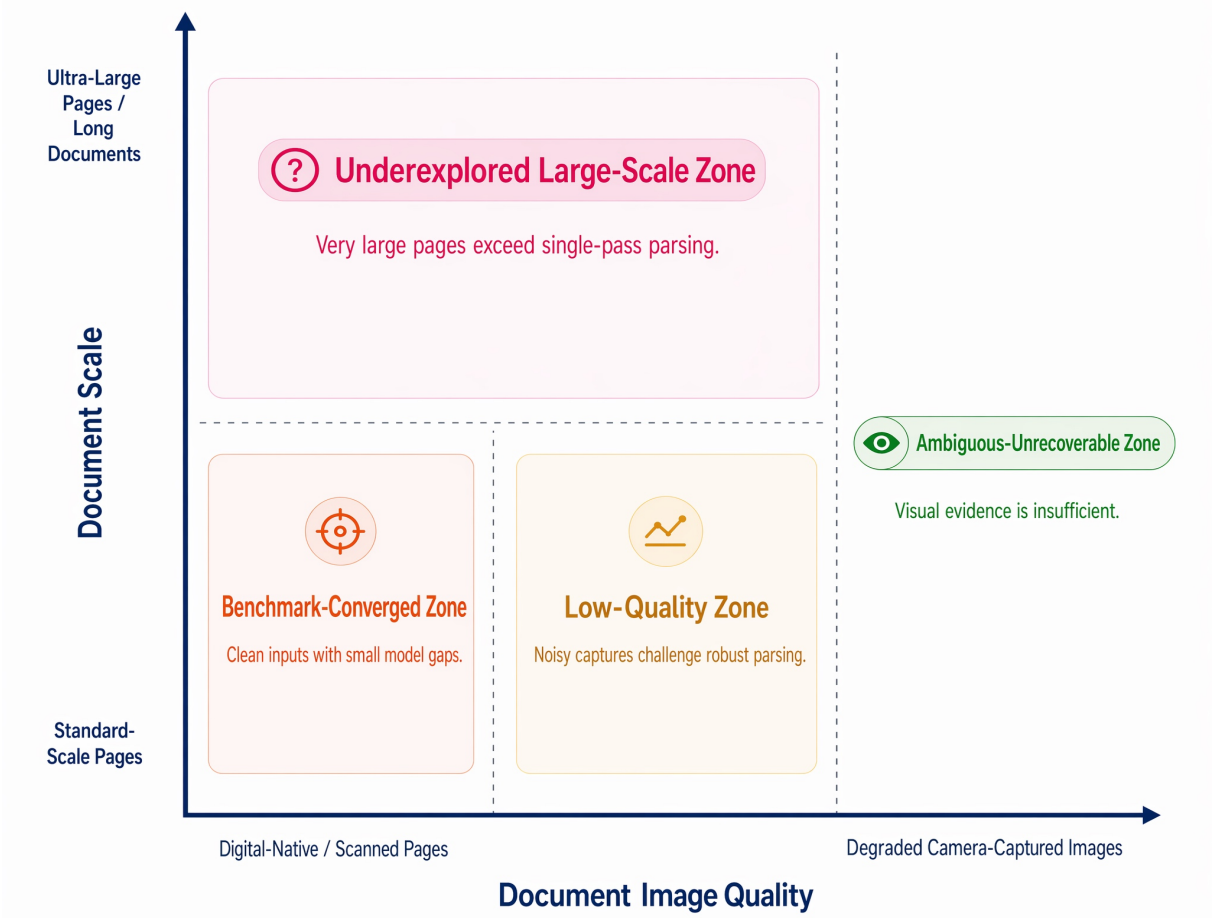

Figure 2: **Document Parsing Capability Matrix.** The horizontal axis represents decreasing document quality, ranging from digital or scanned documents to camera-captured documents affected by perspective distortion, creases, and blur. The vertical axis represents increasing document scale, ranging from single-page or single-image inputs to ultra-long documents and ultra-large images. Based on this matrix, document parsing scenarios can be categorized into the Benchmark-Converged Zone, Low-Quality Zone, Underexplored Large-Scale Zone, and Ambiguous-Unrecoverable Zone.

output token sequences exceed practical model capacity. We argue that this challenge is unlikely to be solved by simply extending the context window of VLMs. Instead, it requires a system-level solution that jointly addresses local content perception, global structural modeling, and cross-region result integration. In this paper, our primary focus is the Low-Quality Zone. For the Underexplored Large-Scale Zone, we present a practical split-then-merge strategy as an initial solution; its design and implementation details are deferred to Appendix A.

Finally, on the far right of the matrix, we identify the **Ambiguous-Unrecoverable Zone**. When document quality becomes extremely poor—beyond reliable visual recognition—current VLMs tend to rely on language priors and contextual regularities to generate speculative predictions [17, 19]. The resulting outputs may appear plausible, yet lack factual grounding in the image itself. While such behavior may occasionally be tolerated in open-ended question answering, it is unacceptable in financial document parsing. Therefore, suppressing hallucinations under low-confidence conditions and adhering to the industrial principle of "better omission than error" remain central challenges for future financial document parsing, especially in risk-sensitive settings where precision is preferred over recall.

To address these challenges, we propose **FinixDoc**, an end-to-end agentic parsing system tailored to financial documents, with **FinixDoc-VL** as its core vision-language model. We build FinixDoc-VL on top of the Qwen3-VL-4B architecture [24]: as discussed above, general-purpose VLMs exhibit stronger robustness than document-specific models in the Low-Quality Zone, while a 4B-scale model offers a practical balance between this robustness and the latency, memory, and cost constraints of real financial deployment. Starting from this backbone, the design of FinixDoc is explicitly organized around the capability zones identified above. For the *Low-Quality Zone*, which is the primary focus of this work, we adapt the visual encoder to financial glyphs through homoglyph-based contrastive learning [26, 10], and

align the model's outputs with business-level requirements through reinforcement learning [31, 6] with composite domain-specific rewards; both stages are further supported by an increased proportion of real-world camera-captured financial documents in training. For the *Underexplored Large-Scale Zone*, we introduce a system-level split-then-merge strategy in the Document Preprocessing Tool Layer, which decomposes ultra-large pages into tractable regions and reconstructs page-level outputs through direction-aware merging. For the *Ambiguous-Unrecoverable Zone*, where confident-yet-ungrounded outputs are unacceptable in financial settings, we adopt the operational principle of "better omission than error" as a design constraint throughout the system, and leave the development of explicit uncertainty-aware refusal mechanisms as future work.

Empirically, FinixDoc-VL substantially improves over the Qwen3-VL-4B base model [24] on OmniDocBench [21], especially on the text and table metrics most relevant to financial documents. More importantly, on the financial-domain benchmark introduced in this work, FinixDoc-VL achieves state-of-the-art performance among major open-source models, with especially large gains in the Low-Quality and Underexplored Large-Scale Zones. These results suggest that progress in real-world financial document parsing should not be judged solely by scores on benchmarks drawn from the Benchmark-Converged Zone. Rather, it should be assessed by whether a system can robustly handle low-quality, high-complexity, and large-scale documents in practical settings.

The main contributions of this paper are summarized as follows:

- We introduce a *Document Parsing Capability Matrix* that characterizes real-world document parsing challenges along the dimensions of document quality and scale, providing a practical framework for analyzing industrial capability boundaries.
- We propose targeted training strategies for financial document parsing, including a ViT adaptation strategy for the financial visual domain and a business-aligned reinforcement learning optimization method, which substantially improve model performance in low-quality scenarios.
- We build an AI-driven *Data Factory* pipeline that supports continuous data expansion and iterative refinement for financial document parsing.
- We introduce **FinixDocBench**, a benchmark closely aligned with real-world financial deployment scenarios, enabling systematic evaluation across digital-native, camera-captured, ultra-large-page, and internal workflow settings. A compliance-reviewed subset of FinixDocBench is released alongside this technical report to support broader research.

2 FinixDoc

2.1 Overall Architecture

FinixDoc is an end-to-end **agentic parsing system** for financial document parsing [38, 28]. Given an input document, the agent leverages a core vision-language model, *FinixDoc-VL*, alongside a suite of deterministic image preprocessing tools to transform raw inputs into robust, page-level visual representations. Rather than executing a static pipeline, the agent dynamically inspects each incoming document and conditionally invokes specific tools based on observed properties, such as input modality, page scale, and image quality indicators.

Supported Inputs and Page-Aligned Outputs

FinixDoc accommodates heterogeneous data types, including office files (Word/PPT/Excel), PDFs, and raster images (PNG/JPG). Across all modalities, the final structured outputs (whether JSON or Markdown) are strictly **page-aligned**, ensuring that the number of output structures perfectly matches the source pages. The agent generates the specific output format (JSON vs. Markdown) according to the format requested in the user's prompt. An illustrative example of the two parsing output formats (JSON and Markdown) generated from `example.pdf` is shown in Fig. 4.

For ultra-large pages, the agent employs a *split-then-merge* strategy: it partitions the oversized page into manageable regions for inference and subsequently stitches the region-level predictions back into a unified result that preserves the original full-page boundaries.

Document Preprocessing Tool Layer

The tool layer provides practical utilities designed to enhance stability and robustness in complex real-world deployments. Typical tools include:

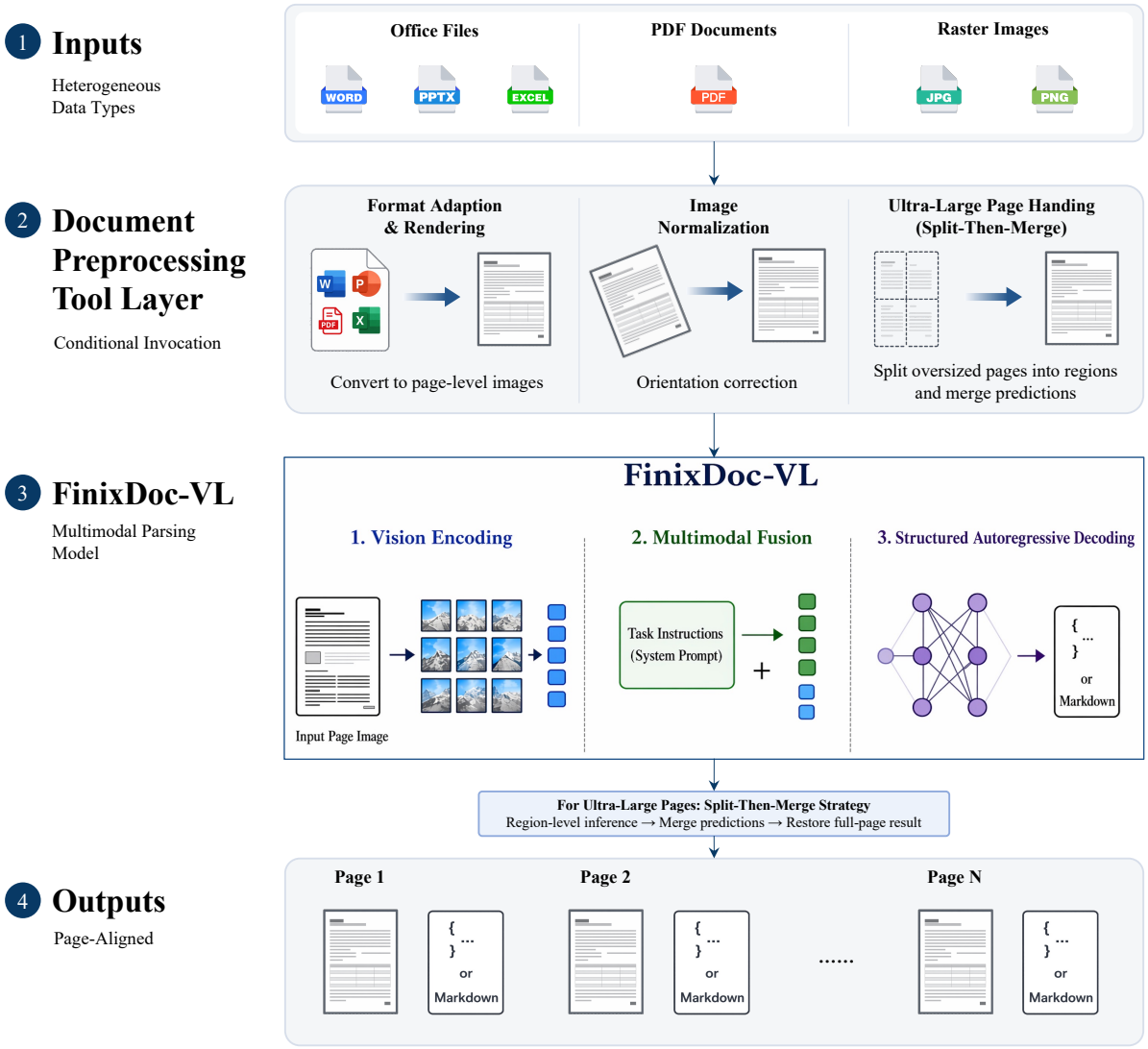

Figure 3: **Overall architecture of FinixDoc. The output format (JSON or Markdown) is specified by the user prompt.**

- **Format Adaptation and Rendering**: Converts office documents and PDFs into page-level images while decoding and normalizing native image inputs.
- **Image Normalization**: Applies lightweight operations, such as scaling and orientation correction (e.g., deskewing or rotation), when quality degradation or misalignment is detected.
- **Ultra-Large Page Handling**: Executes the *split-then-merge* operation when a page exceeds standard visual scales, ensuring the core model processes appropriately sized regions without compromising layout fidelity.

Tool invocation is strictly *conditional*: transformations are applied only when necessary for a specific instance, thereby avoiding unnecessary transformations that could degrade the original document layout. Concretely, image normalization is triggered when lightweight pre-checks detect orientation misalignment or severe scale anomalies, while ultra-large page handling is activated only when the input exceeds the effective visual token budget of FinixDoc-VL or exhibits structural complexity beyond reliable single-pass parsing. The detailed routing criteria for ultra-large page handling are described in Appendix A, which covers token-load estimation and structural-complexity scoring.

FinixDoc-VL

FinixDoc-VL serves as the multimodal core of the parsing system. Starting from the Qwen3-VL-4B backbone [24], FinixDoc-VL is fine-tuned for robust recognition, layout comprehension, and structured generation under low-quality, industrial conditions. Architecturally, it follows the standard VLM inference

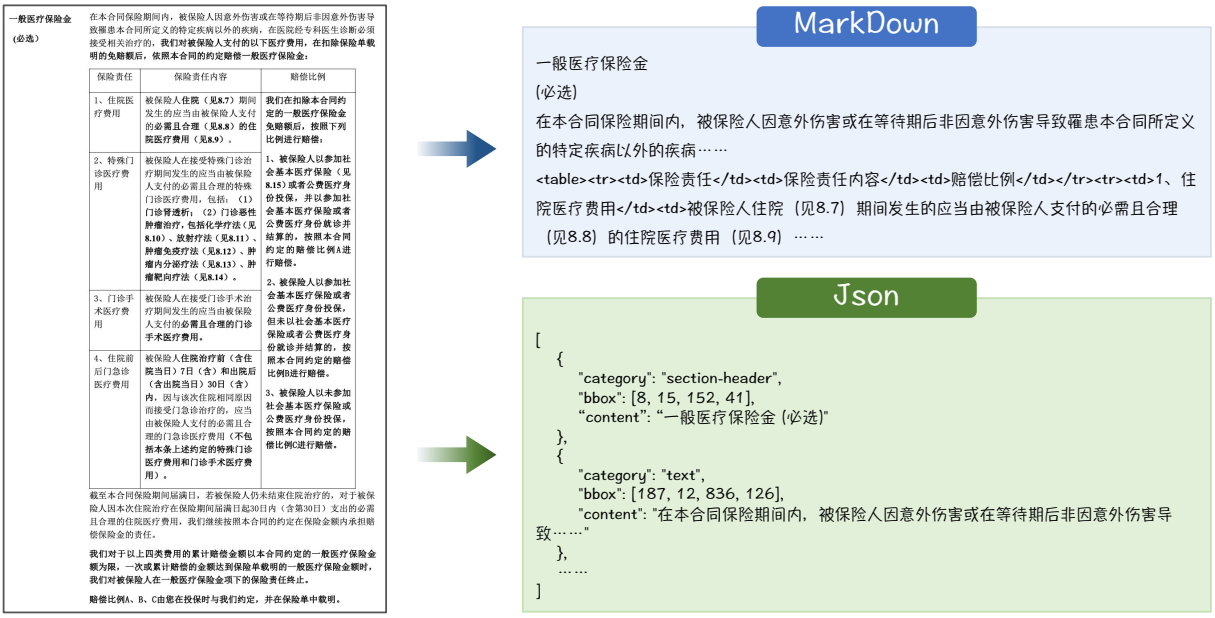

Figure 4: **Parsing output examples on `example.pdf`: Markdown (top) vs. JSON (bottom).**

pipeline of **vision encoding** (mapping the input page image into visual tokens via the vision encoder, followed by spatial merging/compression), **multimodal fusion** (encoding task-specific instructions and fusing them with visual representations into a unified multimodal token sequence), and **structured autoregressive decoding** (generating the final outputs in Markdown or JSON conditioned on the fused representations) [7, 1]. FinixDoc-VL incorporates two domain-specific adaptations: visual representation adaptation in the encoder and business-aligned optimization in the decoder. Both are introduced through the training recipe described in Section 2.2, rather than through architectural modifications. This design enables FinixDoc to retain the end-to-end parsing capability of FinixDoc-VL while also providing practical advantages for real-world deployment. In particular, the system can robustly handle low-quality, camera-captured inputs and can be extended to process ultra-large pages using the *split-then-merge* strategy described above.

2.2 Training Recipe

As indicated by the Low-Quality Zone in the Capability Matrix, general-purpose vision-language models (VLMs) often exhibit stronger robustness than document-specific models when faced with real-world imaging artifacts such as blur, distortion, and shadows. In contrast, document-specific models are often trained on narrower data distributions and are therefore more vulnerable when the input deviates from their training assumptions. However, when adapting general-purpose VLMs to financial document parsing, we observe two major limitations:

1. **Insufficient adaptation of visual representations to the financial domain.**
   The visual encoders used in general-purpose VLMs (e.g., Vision Transformers, ViTs [7]) are typically initialized or adapted from large-scale visual or vision-language pretraining [26, 1]. Real-world financial documents, however, differ substantially from these pre-training sources: they often exhibit denser table structures, stronger layout regularities, and more severe quality degradation introduced during image capture. This mismatch in both domain and difficulty makes it challenging for the model to capture the visual patterns and layout constraints that are specific to financial documents.
2. **Mismatch between existing optimization objectives and financial task requirements.**
   Many document VLMs are still trained primarily with supervised fine-tuning (SFT). Although SFT helps the model imitate the output format and style of the training annotations, it often generalizes poorly to the highly variable and structurally complex cases encountered in financial workflows. Recent work has introduced reinforcement learning (RL) to alleviate some of the limitations of SFT, but the corresponding reward functions are usually designed for general-purpose tasks [22, 30, 31, 6]. Although they address objectives such as instruction following and preference alignment, they do not adequately capture domain-critical information tied to financial semantics, layout structure, and high-risk fields.

To address these issues, we train **FinixDoc-VL** using a two-stage optimization scheme tailored to financial document parsing. Starting from Qwen3-VL-4B as the base model, we retain the general robustness of a strong lightweight VLM while adapting it to the financial domain. In the first stage, we enhance visual and multimodal representations through contrastive learning to improve discrimination of fine-grained financial glyphs and structures. In the second stage, we apply reinforcement learning with domain-specific rewards so that optimization is more closely aligned with the practical requirements of financial document parsing.

2.2.1 Financial Visual Representation Adaptation via Contrastive Learning

To mitigate the domain shift of general visual encoders in financial document scenarios, we propose a contrastive learning approach [26, 10] built on top of the visual branch of Qwen3-VL [24]. The key idea is to construct high-quality hard negatives using visually similar characters (homoglyphs) centered around high-frequency characters in financial documents, and to use them to regularize both the visual representation space and the multimodal generative representation space. This improves the model's ability to discriminate fine-grained glyph differences that are particularly important in financial applications.

Unlike general image–text contrastive learning methods, our approach does not rely on large-scale weakly supervised image–text pairs. Instead, we construct triplets of (*document image, correct text, hard negative text*) tailored to financial document parsing. Importantly, the hard negative texts are not produced by random perturbation. They are generated through homoglyph substitutions derived from real financial document errors, thereby better approximating the error patterns observed in deployment. This stage consists of three components:

1. constructing a financial homoglyph vocabulary and generating hard negative samples accordingly;
2. defining dual-path contrastive objectives over both visual and multimodal generative representations;
3. jointly optimizing the contrastive objectives together with the supervised fine-tuning objective.

The overall training framework is illustrated in Figure 5.

Financial Homoglyph Vocabulary and Negative Sample Construction

We first analyze approximately 100,000 real financial documents and compile a candidate set of more than 4,500 high-frequency characters, including Chinese characters, English letters, digits, and common bilingual symbols. In addition, we build a stroke-order lexicon and a stroke-count lexicon covering more than 20,000 common Chinese characters to facilitate the retrieval of visually similar candidates.

To identify homoglyphs for the 4,500+ high-frequency characters, we design a glyph-similarity scoring module for estimating pairwise similarity between characters. Motivated by the main sources of visual confusion in real financial documents, we characterize character similarity from two major perspectives: rendered-glyph visual similarity and stroke-sequence similarity. The stroke-count signal is further used as a lightweight auxiliary cue during candidate retrieval, but is not used as an independent decisive score in the final filtering stage, since identical stroke counts alone do not necessarily imply visual similarity.

For rendered-glyph visual similarity, we render each character as a grayscale image of fixed size. Given two characters c_i and c_j, with rendered images I_i and I_j, respectively, we compute the structural similarity index [36]:

$$S_{\text{SSIM}}(c_i, c_j) = \text{SSIM}(I_i, I_j). \tag{1}$$

In addition, we compute perceptual hash similarity to capture coarse visual resemblance between rendered glyphs:

$$S_{\text{pHash}}(c_i, c_j) = 1 - \frac{\text{Ham}\left(\text{pHash}(I_i), \text{pHash}(I_j)\right)}{K}, \tag{2}$$

where $\text{Ham}(\cdot,\cdot)$ denotes the Hamming distance and K is the hash length. The rendered-glyph visual similarity is then defined as

$$S_{\text{vis}}(c_i, c_j) = \max\left(S_{\text{SSIM}}(c_i, c_j), S_{\text{pHash}}(c_i, c_j)\right). \tag{3}$$

Rendered glyph images alone are insufficient to fully capture the internal composition of Chinese characters. We therefore introduce stroke-sequence similarity. Let the stroke-order sequences of characters c_i and c_j be

$$\mathbf{s}_i = (s_{i,1}, \ldots, s_{i,m}), \qquad \mathbf{s}_j = (s_{j,1}, \ldots, s_{j,n}). \tag{4}$$

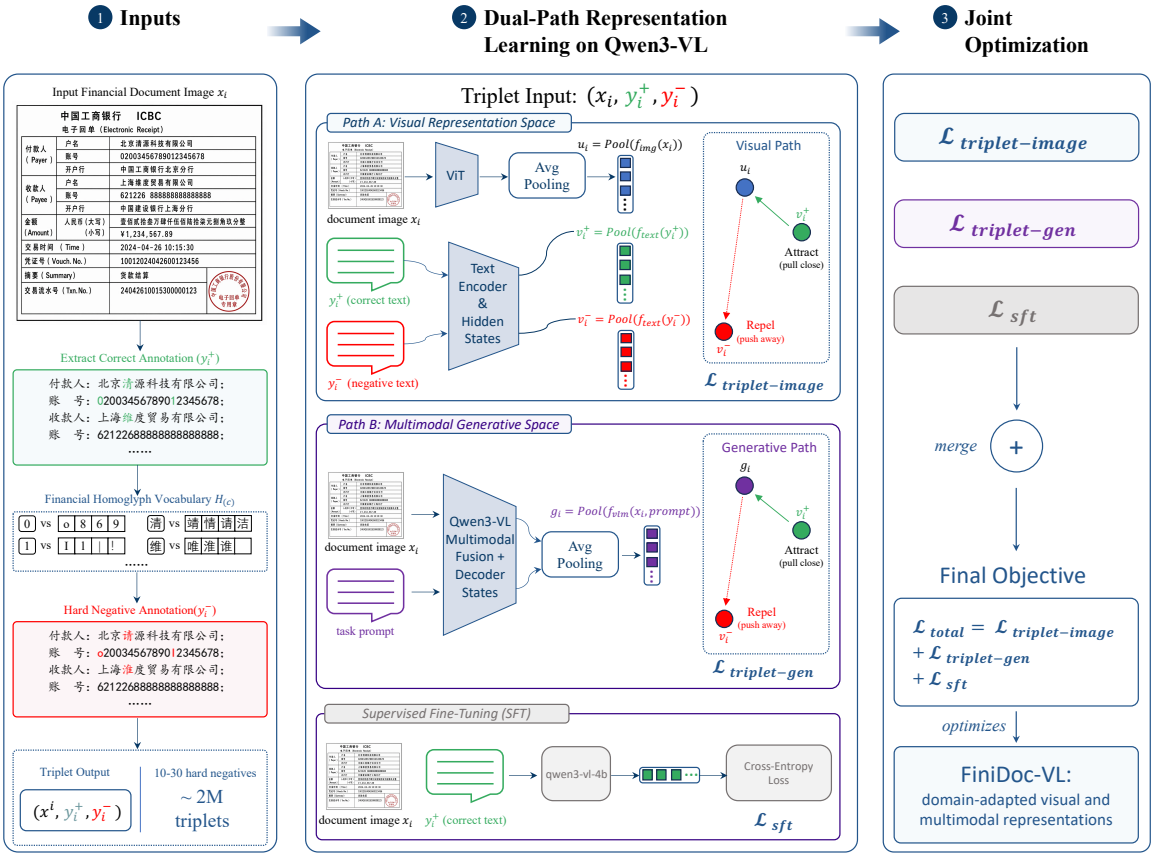

Figure 5: Overall training framework of the proposed financial visual representation adaptation method via contrastive learning. Given a document image and its ground-truth text, hard negative texts are generated through financial homoglyph substitution. The model is then optimized with dual-path contrastive objectives over both visual and multimodal generative representations, jointly with the supervised fine-tuning objective.

We first compute a normalized edit-distance similarity [15]:

$$S_{\text{edit}}(c_i, c_j) = 1 - \frac{\text{EditDist}(\mathbf{s}_i, \mathbf{s}_j)}{\max(m, n)}. \tag{5}$$

To improve robustness to local order shifts and length mismatches, we further combine Dynamic Time Warping (DTW) [27] and Longest Common Subsequence (LCS) signals:

$$S_{\text{order}}(c_i, c_j) = \sigma\left(0.7\left(1 - \frac{\text{DTW}(\mathbf{s}_i, \mathbf{s}_j)}{\max(m, n)}\right) + 0.3\frac{\text{LCS}(\mathbf{s}_i, \mathbf{s}_j)}{\max(m, n)}\right), \tag{6}$$

where $\sigma(\cdot)$ denotes a sigmoid-based normalization function. The stroke-sequence similarity is then defined as

$$S_{\text{stroke}}(c_i, c_j) = \max\left(S_{\text{edit}}(c_i, c_j), S_{\text{order}}(c_i, c_j)\right). \tag{7}$$

Finally, the overall character-level homoglyph similarity score is computed as

$$S_{\text{char}}(c_i, c_j) = \max\left(S_{\text{vis}}(c_i, c_j), S_{\text{stroke}}(c_i, c_j)\right). \tag{8}$$

Equivalently, this aggregation corresponds to taking the maximum over perceptual hash similarity, SSIM similarity, stroke-sequence edit similarity, and DTW–LCS stroke-order similarity. We use the maximum operator because strong similarity under any one reliable view can already lead to practical confusion in financial document recognition, and high recall is preferred when constructing hard-negative candidates.

For candidate retrieval, we additionally use stroke-count similarity as an auxiliary cue for writing complexity. Let the stroke counts of c_i and c_j be n_i and n_j, respectively:

$$S_{\text{count}}(c_i, c_j) = 1 - \frac{|n_i - n_j|}{\max(n_i, n_j)}. \tag{9}$$

This score is used to expand the candidate pool together with the stroke-order lexicon, especially for Chinese characters. However, it is not included in the final maximum aggregation, because similar writing complexity alone may introduce many visually unrelated characters.

Based on this mechanism, we first retrieve more than 200 candidate homoglyphs for each target character using the stroke-order and stroke-count lexicons. We then compute the final similarity score $S_{\text{char}}(c_i, c_j)$ for each candidate pair and retain only those candidates whose score exceeds a fixed threshold of 0.6. This threshold is used consistently in all experiments and empirically provides a practical balance between preserving visually confusable candidates and suppressing low-quality matches. In practice, after thresholding, each target character typically retains 5–30 high-confidence homoglyphs. This range is not manually imposed; rather, it emerges naturally from the score distribution after filtering.

The same overall retrieval-and-filtering framework is applied across Chinese characters, English letters, digits, and common symbols. However, the available similarity cues differ by character type. For Chinese characters, we use both rendered-glyph visual similarity and stroke-sequence similarity, with stroke-count similarity serving as an auxiliary retrieval cue. For English letters, digits, and symbols, rendered-glyph visual similarity serves as the primary criterion, since stroke-order and stroke-count information is either unavailable or less informative for these categories.

Representative examples of the resulting financial homoglyph vocabulary are provided in Appendix B. The vocabulary covers common visual confusions among Chinese characters, English letters, digits, and symbols, and serves as the basis for constructing hard negative samples.

Given a document image x and its correct annotation y^+, we generate hard negatives by replacing selected characters in y^+ with samples drawn from the corresponding homoglyph sets, for example replacing "i" with "|". This produces a hard negative text y^-. Unlike random perturbations, these negatives differ from the correct text at only a small number of visually confusable positions, and therefore better reflect the fine-grained recognition errors observed in financial documents.

This strategy offers two advantages. First, the resulting negative samples are domain-relevant and cover high-risk cases such as amount confusion, name ambiguity, and partial ID-number errors. Second, because the positive and negative texts remain lexically close, the constructed negatives are genuinely hard and thus impose stronger discriminative pressure during training. Using this procedure, we generate 10–30 negative samples for each positive example, yielding approximately 2 million triplets for contrastive learning.

Dual-Path Contrastive Objectives

Our task is ultimately a generation problem: recovering structured text from a document image. Under the VLM framework, this naturally involves at least two representation spaces that benefit from explicit supervision: the visual representation space produced by the image encoder, and the multimodal generative representation space formed after image–text fusion. The former affects perception of local glyphs, layouts, and structural cues, whereas the latter has a more direct influence on the quality of the generated output. Motivated by this observation, we define dual-path contrastive objectives.

For an input image x_i, we obtain patch-level visual representations from the vision encoder and apply average pooling to obtain a global image representation:

$$u_i = \text{Pool}(f_{\text{img}}(x_i)) \in \mathbb{R}^d. \tag{10}$$

For the corresponding correct text y_i^+ and hard negative text y_i^-, we feed them into the language model, take the hidden states from the last layer, and apply average pooling:

$$v_i^+ = \text{Pool}(f_{\text{text}}(y_i^+)), \qquad v_i^- = \text{Pool}(f_{\text{text}}(y_i^-)). \tag{11}$$

It is worth noting that v_i^+ and v_i^- are used as label-side textual prototypes, rather than as direct simulations of the autoregressive decoding trajectory. The purpose of these prototypes is to provide contrastive supervision: image-conditioned representations are encouraged to align with the correct textual prototype and move away from homoglyph-corrupted prototypes. This design allows the contrastive objective to regularize the representation space without modifying the standard supervised fine-tuning path.

Beyond the visual branch, we further extract a multimodal generative representation by feeding the image and the target text into the VLM and pooling the last-layer hidden states:

$$g_i = \text{Pool}(f_{\text{vlm}}(x_i, y_i^+)). \tag{12}$$

Intuitively, g_i summarizes the internal multimodal representation associated with generating the target output from the input image.

Before similarity computation, all representations are ℓ_2-normalized. We then adopt a margin-based contrastive loss with batch-averaged negatives [29, 10]. For an anchor representation z_i, a positive representation p_i, and a set of negative representations $\{n_j\}_{j=1}^N$, we define a learnable scaling factor

$$s = \exp(\texttt{logit\_scale}), \tag{13}$$

and compute

$$\mathcal{L}_{\text{triplet}}(\{z_i\}_{i=1}^N, \{p_i\}_{i=1}^N, \{n_j\}_{j=1}^N) = \frac{1}{N}\sum_{i=1}^{N} \max\left(0,\, s \cdot \frac{1}{N}\sum_{j=1}^{N} z_i^\top n_j - s \cdot z_i^\top p_i + m\right), \tag{14}$$

where N is the batch size and m is a margin hyperparameter, fixed to 0.2 in our experiments.

Based on the dual-path design above, we define one loss on the visual branch and one on the multimodal generative branch:

$$\mathcal{L}_{\text{triplet,img}} = \mathcal{L}_{\text{triplet}}(\{u_i\}_{i=1}^N, \{v_i^+\}_{i=1}^N, \{v_i^-\}_{i=1}^N), \tag{15}$$

$$\mathcal{L}_{\text{triplet,gen}} = \mathcal{L}_{\text{triplet}}(\{g_i\}_{i=1}^N, \{v_i^+\}_{i=1}^N, \{v_i^-\}_{i=1}^N). \tag{16}$$

Here, $\mathcal{L}_{\text{triplet,img}}$ directly regularizes the visual branch, encouraging the encoder to learn fine-grained representations better suited to financial documents. Meanwhile, $\mathcal{L}_{\text{triplet,gen}}$ regularizes the multimodal generative space, improving the model's ability to suppress homoglyph errors during decoding. Together, these two objectives strengthen both visual discrimination and generation-time error resistance.

Joint Optimization with Supervised Fine-Tuning

Contrastive objectives alone are insufficient to preserve the model's sequence generation ability. We therefore jointly optimize the dual-path contrastive losses together with the standard supervised fine-tuning objective. For an input image x and target sequence

$$y = (y_1, \ldots, y_T), \tag{17}$$

the SFT loss is

$$\mathcal{L}_{\text{SFT}} = -\sum_{t=1}^{T} \log P(y_t \mid y_{<t}, x). \tag{18}$$

The final objective is

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_{\text{triplet,img}} + \lambda_2 \mathcal{L}_{\text{triplet,gen}} + \lambda_3 \mathcal{L}_{\text{SFT}}, \tag{19}$$

where $\lambda_1, \lambda_2, \lambda_3$ control the relative weights of the three terms.

Discussion. Unlike standard image–text contrastive learning, our method does not rely on massive weakly supervised image–text pairs. Instead, it models high-risk errors in financial document parsing through task-specific triplets of (*image, correct text, hard negative text*). Compared with random perturbation or generic negative sampling, hard negatives constructed via homoglyph substitution better reflect realistic recognition errors in financial deployments and therefore provide more informative training signals. Starting from Qwen3-VL pre-trained weights, this stage gradually adapts both the visual branch and the multimodal representation space while retaining the base model's general capabilities, thereby establishing a stronger representation foundation for subsequent reinforcement learning with domain-specific rewards.

2.2.2 Reinforcement Learning Fine-Tuning with Domain-Specific Rewards

After enhancing the model's visual and multimodal representations, we further optimize FinixDoc-VL end-to-end with reinforcement learning, shifting the objective from pure imitation of annotated outputs toward direct optimization of task-relevant utility. In financial document parsing, token-level cross-entropy alone has clear limitations. High-risk errors are not uniformly distributed across tokens; instead, they concentrate in critical elements such as amounts, dates, ID numbers, field names, table structures, and cross-region reading order. These factors are not adequately captured by generic text-similarity metrics, and a single supervised objective is insufficient to express the trade-offs among them.

We therefore adopt a GRPO-based reinforcement learning approach [31, 6] and design domain-specific reward functions tailored to financial document parsing. Furthermore, because text quality, detection quality, and reading order exhibit different reward characteristics and optimization dynamics, we employ a multi-stage RL strategy to improve training stability and overall alignment.

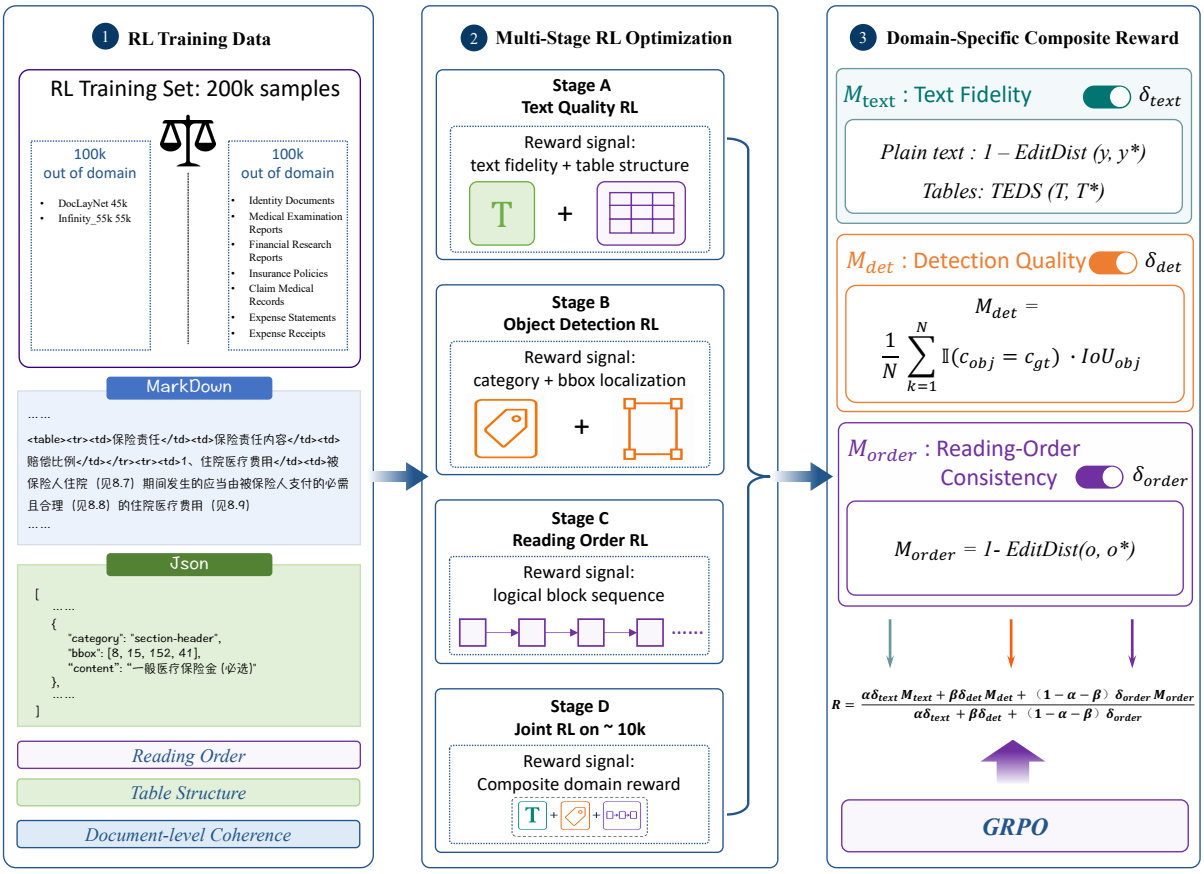

Figure 6: Overview of reinforcement learning fine-tuning with domain-specific rewards. The RL stage jointly optimizes JSON and Markdown generation through multi-stage sub-capability calibration followed by unified GRPO alignment. Task-dependent reward masks combine text fidelity, detection quality, and reading-order consistency according to the output format.

As summarized in Figure 6, the RL stage starts from dual task formulations over JSON and Markdown generation, proceeds through multi-stage sub-capability calibration, and finally performs unified GRPO alignment using task-dependent domain rewards.

Training Data and Task Formulation

Our RL training set contains 200k samples, including 100k out-of-domain samples and 100k in-domain financial samples. The out-of-domain portion is drawn primarily from public document parsing datasets and is used to preserve general layout understanding and open-domain structural generalization [23, 42, 16, 35]. The in-domain portion covers common and high-value document types in real financial workflows and is used to strengthen performance on complex layouts, low-quality images, and high-risk fields. The detailed composition is shown in Table 1.

During RL, we optimize two generation tasks jointly. The first is structured JSON generation. Given an input page, the model outputs a sequence of structural objects, each containing fields such as category, bounding box, and text content. This task evaluates layout detection, category prediction, localization, and consistency of structured outputs. The second is Markdown generation. Given an input page, the model generates the full page content in natural reading order, recovering paragraph structure, heading hierarchy, and table content as faithfully as possible, with tables represented in an HTML-compatible format. Compared with the JSON task, Markdown generation places greater emphasis on global textual coherence, reading-order reconstruction, and faithful recovery of complex tables.

These two tasks correspond to complementary usage patterns in practical systems. JSON output is well suited as an intermediate parsing layer for downstream rule engines, field extraction modules, and structured business systems. Markdown output is more suitable as a unified document representation for downstream LLM-based tasks such as question answering, summarization, and clause understanding. By optimizing both tasks, we encourage the model to acquire both fine-grained structural parsing ability and stronger document-level understanding.

Table 1: Composition of the training data used in the reinforcement learning stage.

Source	Dataset / Document Type	Size
Out-of-domain	DocLayNet	45k
Out-of-domain	Infinity-Doc-55K	55k
In-domain	Identity Documents	20k
In-domain	Medical Examination Reports	10k
In-domain	Financial Research Reports	10k
In-domain	Insurance Policies	10k
In-domain	Claim Medical Records	10k
In-domain	Expense Statements	20k
In-domain	Expense Receipts	20k
Total		200k

Multi-Stage Optimization Strategy

We do not optimize all RL objectives jointly from the beginning. Instead, we adopt a multi-stage pipeline, which can be summarized as *sub-capability calibration followed by unified alignment*. Specifically, we first optimize the three sub-objectives—text quality, object detection, and reading order—in separate stages, allowing the model to stabilize under more focused reward signals. We then perform joint RL on approximately 10k high-quality samples using a unified composite reward to achieve final alignment across capabilities.

This strategy is motivated by two considerations. First, heterogeneous rewards can interfere strongly during early RL training; for example, the model may sacrifice localization quality to increase text similarity, or degrade reading order to improve detection consistency. Multi-stage training helps reduce such instabilities. Second, the final joint stage explicitly teaches the model to balance multiple objectives, so that it can maintain text fidelity, structural integrity, localization quality, and reading-order consistency simultaneously at inference time. We have found this strategy to be particularly important for complex financial documents such as long tables, mixed-layout pages, and insurance clauses.

GRPO Optimization Objective

We adopt Group Relative Policy Optimization (GRPO) as the policy optimization algorithm [31, 6]. For an input sample x, the current policy π_θ generates a group of candidate outputs $\{y_i\}_{i=1}^{G}$, and each candidate is assigned a reward $R(x, y_i)$. We normalize the rewards within the group to obtain relative advantages:

$$A_i = \frac{R(x, y_i) - \mu_R}{\sigma_R + \varepsilon_R}, \tag{20}$$

where μ_R and σ_R are the mean and standard deviation of the group rewards, respectively, and ε_R is a small constant for numerical stability. The GRPO objective is

$$\mathcal{L}_{\text{GRPO}} = -\mathbb{E}_{x,\{y_i\}} \left[\frac{1}{G} \sum_{i=1}^{G} \min \left(r_i A_i,\ \text{clip}(r_i, 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}}) A_i \right) \right] + \lambda_{\text{KL}} D_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}}), \tag{21}$$

where

$$r_i = \frac{\pi_\theta(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}. \tag{22}$$

Here, π_{ref} is a reference policy, ϵ_{clip} is the clipping coefficient, and λ_{KL} controls the KL regularization strength. The KL term prevents the updated policy from drifting excessively during RL training. Compared with standard PPO formulations that often rely on explicit value estimation, GRPO constructs policy gradients directly from relative rewards within a sampled group. This is particularly suitable for structured document parsing, where the output space is complex and the reward naturally combines multiple heterogeneous metrics.

Domain-Specific Reward Design

To improve the practical utility of structured document parsing, we design a composite reward that jointly evaluates text fidelity, detection quality, and reading-order consistency. Because not every reward component applies to every output format, we introduce task-dependent binary masks $\delta_{\text{text}}, \delta_{\text{det}}, \delta_{\text{order}} \in \{0, 1\}$ and define

$$R = \frac{\alpha\delta_{\text{text}} M_{\text{text}} + \beta\delta_{\text{det}} M_{\text{det}} + (1-\alpha-\beta)\delta_{\text{order}} M_{\text{order}}}{\alpha\delta_{\text{text}} + \beta\delta_{\text{det}} + (1-\alpha-\beta)\delta_{\text{order}}}, \tag{23}$$

where M_{text}, M_{det}, and M_{order} denote the rewards for text fidelity, detection quality, and reading-order consistency, respectively. The hyperparameters $\alpha, \beta \in [0, 1]$ satisfy $\alpha + \beta \leq 1$. For inactive reward terms, the corresponding mask is set to 0, so unavailable components are excluded and the remaining weights are renormalized automatically. In practice, at least one reward term is active for every task, so the denominator is always non-zero. Table 2 summarizes the actual hyperparameter values and mask settings used in our training.

Table 2: Reward hyperparameters and mask settings for the JSON and Markdown tasks.

Task	α	β	$1-\alpha-\beta$	δ_{text}	δ_{det}	δ_{order}
JSON	0.5	0.4	0.1	1	1	1
Markdown	0.5	0.4	0.1	1	0	1

The text-fidelity reward measures similarity between the prediction and the ground truth over both plain-text and table objects. For regular text objects, we use normalized edit distance [15]:

$$M_{\text{text,plain}} = 1 - \text{EditDist}(y, y^*), \tag{24}$$

where y and y^* are the predicted and reference text, respectively, and $\text{EditDist}(\cdot, \cdot) \in [0, 1]$ denotes normalized edit distance. For table objects, character-level similarity alone is insufficient because a table may contain mostly correct cell content while still breaking row–column relations or hierarchical structure. We therefore use Tree-Edit-Distance-based Similarity (TEDS) [40, 43]:

$$M_{\text{text,table}} = \text{TEDS}(T, T^*), \tag{25}$$

where T and T^* are the predicted and reference table structures, respectively. The overall text reward is then computed by weighted aggregation over all page objects:

$$M_{\text{text}} = \frac{\sum_k w_k M_k}{\sum_k w_k}, \tag{26}$$

where M_k is instantiated as either $M_{\text{text,plain}}$ or $M_{\text{text,table}}$ depending on the object type, and w_k denotes the importance weight of the k-th object.

The detection reward is used primarily for JSON outputs, where the model must predict both structural categories and bounding boxes. Let prediction–ground-truth object pairs be matched by a predefined assignment rule.[1] Let $\mathcal{P}$ denote the set of matched pairs, with $N_{\text{match}} = |\mathcal{P}|$. For each matched pair $(\hat{o}, o^*) \in \mathcal{P}$, let $c_{\hat{o}}$ and c_{o^*} be the predicted and ground-truth categories, and let $\text{IoU}(\hat{o}, o^*)$ be the intersection-over-union of the corresponding boxes. We define

$$M_{\text{det}} = \frac{1}{N_{\text{match}}} \sum_{(\hat{o}, o^*) \in \mathcal{P}} \mathbb{I}\{c_{\hat{o}} = c_{o^*}\} \cdot \text{IoU}(\hat{o}, o^*), \tag{27}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function. This reward encourages the model to produce correct structural labels together with accurate localization.

The reading-order reward evaluates whether the model preserves the logical reading sequence of the document, which is particularly important for multi-column pages, mixed text–table layouts, and long documents. Let o denote the predicted object or block sequence and o^* the corresponding reference order. For Markdown outputs, this sequence is derived from the linearized block order in the rendered structure. We define

$$M_{\text{order}} = 1 - \text{EditDist}(o, o^*), \tag{28}$$

where $\text{EditDist}(\cdot, \cdot)$ is normalized to $[0, 1]$. This reward explicitly constrains document-level ordering consistency and penalizes outputs whose local content is correct but whose global reading order is inconsistent.

[1] In implementation, matching can be performed by maximum-IoU assignment or Hungarian matching, depending on the annotation format.

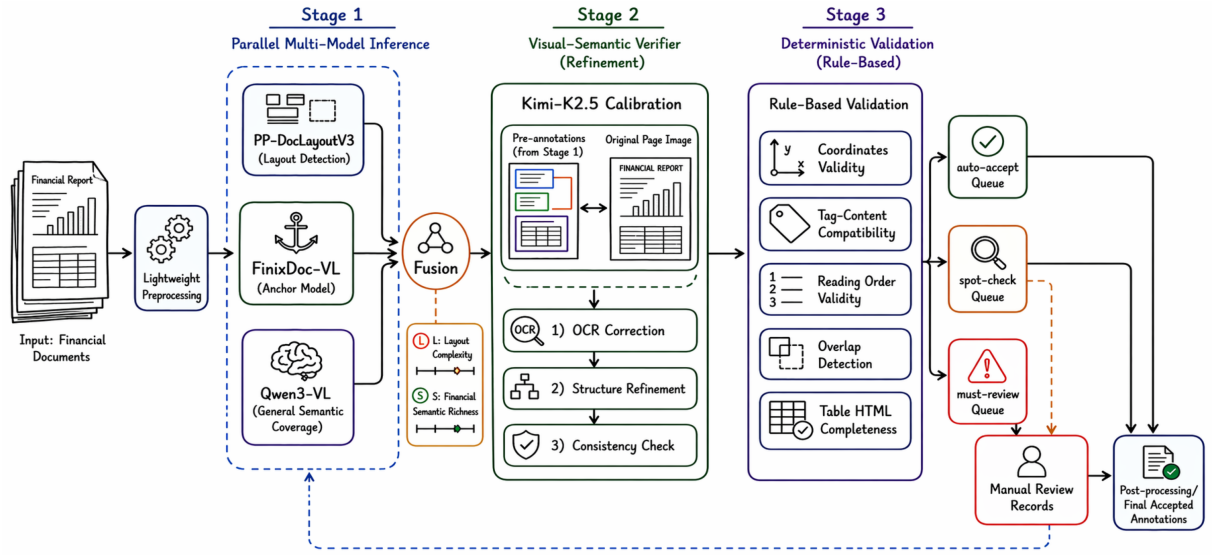

Figure 7: **Overview of the Data Factory pipeline.**

Discussion. Our reward design is task-dependent. For JSON outputs, text fidelity, structural category, and localization are combined at the object level before page-level aggregation. For Markdown outputs, the reward emphasizes text fidelity, reading-order consistency, and structural preservation of tables through TEDS. Unlike RL approaches based on generic text-level metrics such as BLEU, ROUGE, or global edit distance, our method incorporates objectives that directly reflect the requirements of practical financial document parsing. In particular, the model is encouraged to produce accurate text, correct structural categories, precise bounding boxes, coherent reading order, and structurally faithful tables. This makes the RL stage a useful complement to SFT for improving structural consistency and practical usability.

3 Data Factory: A Human-in-the-Loop Production Pipeline for Financial Documents

3.1 Motivation

High-quality structured data is fundamental to training and improving document parsing models. In the financial and insurance domain, however, data construction faces three major practical challenges.

First, **high domain specificity**: annotation requires accurate handling of financial terminology, clause hierarchies, and highly specialized document layouts.

Second, **large quality variation**: a single production pipeline must remain compatible with digitally native PDFs, scanned copies, and mobile-captured images of varying quality.

Third, **massive data scale**: historical business operations have accumulated corpora at the scale of millions of financial documents and tens of millions of document pages, while large-scale manual annotation remains prohibitively expensive.

To our knowledge, few existing industrial pipelines jointly address these three challenges in a unified design. Many focus on a single aspect of the problem, such as hard-example mining, multi-model agreement, or progressive annotation, without integrating model diversity, multi-stage refinement, and downstream quality feedback into a closed loop. To address this gap, we build an AI-driven, human-in-the-loop *Data Factory* pipeline for producing structured training and evaluation data from large-scale raw financial documents. The system adopts a three-stage cascaded architecture that combines heterogeneous multi-model inference, model-based refinement, and rule-based validation, followed by confidence-aware routing for automatic acceptance, spot-checking, or mandatory expert review. In practice, this design shifts human effort from full manual annotation to targeted auditing and correction of high-risk samples, making large-scale data production more feasible.

3.2 Pipeline Architecture and Core Mechanisms

The Data Factory consists of lightweight preprocessing, a three-stage core pipeline, and post-processing. The main architecture and operating mechanisms are summarized below.

Stage 1: Heterogeneous Multi-Model Collaborative Inference and Multi-Dimensional Scoring

Each page first enters a parallel multi-model inference framework to extract both layout structures and semantic content. Within this framework, we utilize three distinct models: PP-DocLayoutV3, which is dedicated exclusively to stable layout detection without extracting textual content; FinixDoc-VL, which provides financial-domain parsing and schema-aligned structural priors; and Qwen3-VL-235B-A22B-Instruct, a large-scale general-purpose VLM that complements domain-specific priors with broader semantic coverage [33, 24]. Since FinixDoc-VL is the model most closely aligned with our target domain and annotation schema, its output is used as the anchor in the subsequent fusion stage. Specifically, objects proposed by the other models are matched to the FinixDoc-VL anchor output based on category compatibility, spatial overlap, and text similarity. Model-specific confidence and agreement signals are then aggregated to produce a unified candidate annotation for the current page.

Based on the comprehensive parsing results obtained from this multi-model fusion, the system evaluates each page by computing two page-level statistics: layout complexity $\mathcal{L}$ and financial semantic richness $\mathcal{S}$. Let the fused page-level annotation be denoted as

$$\mathcal{B} = \left\{ b_i = (x_i^{(1)}, y_i^{(1)}, x_i^{(2)}, y_i^{(2)}, c_i, t_i) \right\}_{i=1}^{n}, \tag{29}$$

where b_i denotes the i-th fused layout region, $(x_i^{(1)}, y_i^{(1)}, x_i^{(2)}, y_i^{(2)})$ is its normalized bounding box, c_i is its layout category, and t_i is the extracted textual content. We further define the geometric region of b_i as

$$r_i = [x_i^{(1)}, x_i^{(2)}] \times [y_i^{(1)}, y_i^{(2)}], \tag{30}$$

and its normalized area as

$$a_i = |r_i| = (x_i^{(2)} - x_i^{(1)})(y_i^{(2)} - y_i^{(1)}). \tag{31}$$

The layout complexity score $\mathcal{L}$ measures structural difficulty by considering layout region count, layout density, overlap complexity, and layout-type diversity. Let s_{reg}, s_{den}, s_{ovl}, and s_{div} denote the corresponding normalized component scores. We compute $\mathcal{L}$ as

$$\mathcal{L} = 0.30\, s_{\text{reg}} + 0.25\, s_{\text{den}} + 0.20\, s_{\text{ovl}} + 0.25\, s_{\text{div}}. \tag{32}$$

The normalized region count score is defined as

$$s_{\text{reg}} = \min\left(\frac{n}{N_{\max}}, 1 \right), \tag{33}$$

where $N_{\max}$ is the reference maximum number of layout regions per page.

The layout density score is defined as the union area covered by all fused layout regions:

$$s_{\text{den}} = \left| \bigcup_{i=1}^{n} r_i \right|. \tag{34}$$

Since all page coordinates are normalized to $[0, 1]$, this value naturally lies within $[0, 1]$.

The overlap complexity score measures the degree of spatial conflict among layout regions:

$$s_{\text{ovl}} = \begin{cases} 0, & n < 2, \\ \min\left(\frac{\sum_{i<j} |r_i \cap r_j|}{\tau_{\text{o}} \sum_{i=1}^{n} a_i + \epsilon}, 1 \right), & n \geq 2, \end{cases} \tag{35}$$

where τ_{o} is a normalization constant and ϵ is a small constant used to avoid division by zero.

The layout-type diversity score is defined as the fraction of predefined layout categories that appear on the page:

$$s_{\text{div}} = \frac{|\{c_i \mid b_i \in \mathcal{B}\}|}{|\mathcal{C}|}, \tag{36}$$

where $\mathcal{C}$ denotes the predefined set of layout categories.

Meanwhile, the financial semantic richness score $\mathcal{S}$ measures semantic informativeness based on the extracted text, including OCR text volume, financial term density, and content diversity. Let

$$\mathcal{T} = \{b_i \in \mathcal{B} \mid |t_i| > 0\} \tag{37}$$

be the set of regions with non-empty textual content, let $\mathcal{F}$ denote a predefined financial-domain term dictionary, and let $\mathcal{C}_{\text{text}} \subseteq \mathcal{C}$ denote the subset of layout categories that can contain textual information.

Let s_{text}, s_{fin}, and s_{cont} denote the normalized OCR text volume, financial-term density, and content diversity scores, respectively. The semantic richness score is computed as

$$\mathcal{S} = 0.35\, s_{\text{text}} + 0.40\, s_{\text{fin}} + 0.25\, s_{\text{cont}}. \tag{38}$$

The OCR text volume score is normalized by a reference maximum text length $T_{\max}$:

$$s_{\text{text}} = \min\left(\frac{\sum_{b_i \in \mathcal{T}} |t_i|}{T_{\max}}, 1\right). \tag{39}$$

The financial-term density score is defined as the normalized proportion of financial-domain terms in the extracted text:

$$s_{\text{fin}} = \min\left(\frac{1}{\rho_{\text{f}}} \cdot \frac{\sum_{w \in \text{tokens}(\mathcal{T})} \mathbb{I}\{w \in \mathcal{F}\}}{|\,\text{tokens}(\mathcal{T})| + \epsilon}, 1\right), \tag{40}$$

where ρ_{f} is the reference financial-term density. In our implementation, we set $\rho_{\text{f}} = 0.10$, meaning that a page whose financial-domain terms account for 10% of all extracted tokens reaches the maximum normalized financial-term density.

The content diversity score is defined as the fraction of text-bearing layout categories present on the page:

$$s_{\text{cont}} = \frac{|\{c_i \mid b_i \in \mathcal{T}\}|}{|\mathcal{C}_{\text{text}}|}. \tag{41}$$

Finally, these two heuristic composite scores are used for post-hoc data selection and stratified sampling. The overall page value score, denoted by V_{page}, is defined as

$$V_{\text{page}} = 0.5\, \mathcal{L} + 0.5\, \mathcal{S}. \tag{42}$$

Pages are assigned to four sampling quadrants using thresholds $\theta_{\mathcal{L}}$ and $\theta_{\mathcal{S}}$. A page is regarded as high-complexity if $\mathcal{L} \geq \theta_{\mathcal{L}}$ and high-semantics if $\mathcal{S} \geq \theta_{\mathcal{S}}$. These two binary decisions form the four-quadrant sampling space used in the subsequent data selection stage.

Stage 2: Cascaded Large-Model Calibration and Refinement

After the multi-model fusion stage produces the initial candidate annotations, these outputs are not directly treated as final labels. Instead, the fused annotations are passed, together with the original page image and the corresponding extracted textual evidence, into a cascaded refinement pipeline driven by a more powerful multimodal foundation model, Kimi-K2.5 [12]. In this stage, Kimi-K2.5 serves as a visual-semantic verifier and refiner that re-examines the pre-annotated results against the original document image, the recognized text, and the page-level structural context. This design allows the system to correct residual errors introduced during heterogeneous model collaboration, such as cross-model disagreement, noisy object proposals, inaccurate bounding boxes, OCR inconsistencies, duplicated regions, and hallucinated layout elements.

The refinement process is organized as a three-step cascade, where each step focuses on a specific class of annotation errors while preserving the outputs of previous stages for traceability:

- **Step 1 (OCR):** performs low-level textual verification and correction. The model compares the pre-annotated textual content with the visual evidence in the original page, revising OCR errors, recovering missing or truncated text, and correcting basic category misclassifications when the textual pattern clearly conflicts with the assigned label.
- **Step 2 (Struct):** focuses on structural calibration of the annotation layout. Kimi-K2.5 verifies whether each bounding box accurately covers the intended visual region, adjusts misaligned or fragmented boxes, removes duplicated proposals, and optimizes the reading order according to the spatial organization of the page. For complex table pages, this step activates a table integrity protection mechanism to avoid breaking coherent table structures during box adjustment and order refinement.

- **Step 3 (Consist):** conducts final global consistency checking across the refined annotations. The model inspects whether the page-level annotation set is coherent with the original image and with the outputs accumulated from earlier stages, mitigating common VLM failure modes such as hallucinated layout elements, unsupported semantic assignments, missing key regions, and unresolved cross-model inconsistencies.

By introducing this larger-model calibration stage after the initial heterogeneous fusion, the pipeline separates broad candidate generation from high-precision annotation correction. The first stage maximizes recall by aggregating complementary model outputs, while the second stage uses Kimi-K2.5 to improve precision and structural reliability through image-grounded refinement. All intermediate annotations and revision records are preserved, enabling end-to-end traceability and providing explicit evidence for the subsequent validation and quality routing stage.

Stage 3: Structured Validation and Quality Feedback Loop

After model refinement, the annotations enter a programmatic validation stage driven by deterministic rules. The system performs structured checks and lightweight rule-based repairs along multiple dimensions, including coordinate validity, tag-content compatibility, reading-order consistency, box-overlap anomalies, and table HTML completeness. Based on the validation results, the system aggregates multiple confidence signals, such as inter-model agreement, refinement stability across stages, OCR consistency, and rule-violation counts, and routes each page into one of three queues: `auto-accept`, `spot-check`, and `must-review`. Pages routed to `spot-check` are manually audited by sampling, while pages routed to `must-review` undergo mandatory human review and correction before final acceptance.

The resulting human inspection records, including corrected annotations, error categories, and reviewer comments, are then fed back into the Stage 1 pre-annotation process. In subsequent production rounds, these feedback signals are used to update annotation guidelines, refine model prompts, adjust fusion heuristics, and calibrate confidence thresholds, thereby improving the precision of initial candidate annotations. This feedback mechanism allows human review experience to continuously improve upstream pre-annotation quality, while the confidence-aware routing mechanism concentrates costly human inspection on high-risk samples where model uncertainty remains high.

3.3 Annotation Standards and Domain-Specific Ambiguity Resolution

Reliable large-scale data production requires a stable annotation standard. Our annotation guidelines were jointly developed by domain experts and the annotation team and were designed to remain compatible with the open-source MinerU 2.5 framework [20]. The entire pipeline uses 10 standard page-element tags: `page-header`, `page-footer`, `title`, `section-header`, `text`, `table`, `picture`, `caption`, `footnote`, and `other`.

To better match the characteristics of financial documents, we do not treat `list-item` and `formula` as standalone classes. In particular, standalone mathematical expressions occur only sporadically across financial documents, accounting for a negligible fraction of all layout elements; treating them as a dedicated category therefore introduces schema overhead without commensurate benefit. Accordingly, list items are categorized semantically as either `text` or `section-header`, while mathematical expressions are serialized as inline LaTeX spans within the `text` category rather than being annotated as a separate layout class. This design reduces schema fragmentation and better matches the dominant usage patterns in financial documents. In addition, the Data Factory includes dedicated rules for several high-frequency ambiguity cases commonly observed in financial documents:

- **Boundary between hierarchical headings (`section-header`) and body text (`text`):** if a text span begins with hierarchical numbering (e.g., "I.", "1.1", "(1)") and functions as a discourse-level section introducer in context, it is classified as `section-header`. By contrast, the detailed provisions listed under such numbering are classified as `text`.
- **Distinguishing figure/table descriptions (`caption`) from footnotes (`footnote`):** a `caption` must be adjacent to the figure or table that it describes, whereas text placed at the bottom of the page and serving as a page-level disclaimer or supplementary note is classified as `footnote`.
- **Table integrity protection (`table`):** for complex tables with many merged cells, such as insurance premium rate tables and cash value tables, the system prevents a single logical table from being split into multiple independent `text` regions. Instead, a dedicated table-preservation mechanism is applied, and the table is serialized uniformly as HTML `<table>` content that preserves merged-cell structure.

For geometric representation, the entire pipeline uses normalized coordinates $[x_1, y_1, x_2, y_2] \in [0, 1]^4$,

with denormalization performed only at final export. For all elements except `picture`, the `content` field retains the associated text, structural markers, or serialized content (e.g., Markdown heading markers such as #, ##, or HTML table content).

3.4 Data Sampling Strategy and Final Output

The Data Factory is initialized with a pool of approximately 10 million raw financial document pages. After the three-stage pipeline and confidence-aware routing, pages are retained in the validated page pool only if they are either automatically accepted with sufficient confidence or accepted after the applicable spot-check or mandatory human-review procedure. Instead of applying random sampling to this validated pool, we perform **four-quadrant stratified sampling** based on the layout complexity $\mathcal{L}$ and financial semantic richness $\mathcal{S}$ computed in Stage 1:

- **Q1 (High Complexity + High Semantics):** fully retained as the core backbone set for downstream training, especially reinforcement learning.
- **Q2 (Low Complexity + High Semantics):** up-sampled to increase coverage of long-tail financial semantics.
- **Q3 (High Complexity + Low Semantics):** not directly used in standard training; instead, reserved as a candidate pool for robustness-oriented training, such as error-correction training and hard-case augmentation.
- **Q4 (Low Complexity + Low Semantics):** heavily down-sampled or filtered out.

Through this quality-aware data selection strategy, we ultimately retain approximately 100,000 pages of validated in-domain training data, corresponding to an overall retention rate of roughly 1% at the page level. This curated distribution helps reduce overfitting to a narrow set of document layout patterns during training.

The final in-domain dataset covers major high-frequency scenarios in financial workflows, including Identity Documents, Medical Examination Reports, Financial Research Reports, Insurance Policies, Claim Medical Records, Expense Statements, and Expense Receipts. In the subsequent reinforcement learning (RL) stage, these 100,000 pages of refined in-domain data are mixed with an additional approximately 100,000 pages of out-of-domain public data, such as DocLayNet and Infinity-Doc-55K, resulting in a final training set of roughly 200,000 pages [23, 35]. This data mixture improves both cross-domain generalization and in-domain robustness for financial document parsing.

4 Evaluation

4.1 Evaluation Benchmarks

We evaluate our core model, FinixDoc-VL, against representative document parsing baselines on two benchmarks: OmniDocBench v1.5 and our proposed FinixDocBench. For the ultra-large-page subset of FinixDocBench, we further evaluate the complete FinixDoc system, including the Document Preprocessing Tool Layer and its split-then-merge (tiling-and-merging) module. Together, these evaluations cover a broad spectrum of settings, ranging from general document parsing to highly specialized and deployment-critical financial scenarios.

4.1.1 OmniDocBench v1.5

OmniDocBench v1.5 is a public benchmark for general document parsing [21]. It contains 1,355 pages spanning diverse document types, including academic papers, technical reports, and commercial contracts, with an average of over 1,100 tokens per page. The benchmark covers common layout patterns such as multi-column text, complex tables, mathematical formulas, and nested document structures. It primarily evaluates foundational parsing capabilities under relatively clean, digitally native conditions.

The official overall score is defined as:

$$S_{\text{ODB}} = \frac{100(1 - E_{\text{text}}) + \text{TEDS}_{\text{table}} + \text{CDM}_{\text{formula}}}{3}, \tag{43}$$

where E_{text}, $\text{TEDS}_{\text{table}}$, and $\text{CDM}_{\text{formula}}$ denote the text edit score, the table TEDS score, and the formula CDM score, respectively.

Although FinixDoc-VL is not explicitly optimized for mathematical formula parsing, we still evaluate the formula task on OmniDocBench v1.5 to provide a complete and fair comparison. By contrast, some

compared baselines report formula results while others do not; therefore, we use their official overall scores where available. To enable a more comparable evaluation on the shared dimensions, we further introduce an auxiliary metric that excludes the formula component, denoted as $S_{\text{ODB}}^{\text{wo-formula}}$:

$$S_{\text{ODB}}^{\text{wo-formula}} = \frac{100(1 - E_{\text{text}}) + \text{TEDS}_{\text{table}}}{2}. \tag{44}$$

In the following analysis on OmniDocBench v1.5, we use $S_{\text{ODB}}^{\text{wo-formula}}$ as the primary comparison metric. This choice is also consistent with the industrial focus of our work: mathematical formulas account for only a negligible fraction of real-world financial documents and rarely constitute the practical bottleneck in downstream deployment.

4.1.2 FinixDocBench

As discussed in the Introduction, the main challenges in real-world financial document parsing arise from two sources: low-quality camera-captured inputs and complex large-scale pages, such as ultra-long documents and dense large tables. Existing public benchmarks provide only limited coverage of these challenging settings, leaving a non-trivial gap between benchmark performance and real deployment effectiveness.

To bridge this gap, we construct **FinixDocBench**, a benchmark specifically designed for financial document parsing. It systematically evaluates model capabilities on digitally native financial documents, camera-captured documents, and ultra-large pages. Compared with existing public benchmarks, FinixDocBench has three distinguishing characteristics:

1. **Comprehensive quality coverage:** it includes both digitally native documents and authentically noisy camera-captured images;
2. **Broad scale variation:** it spans standard pages, dense large tables, and ultra-long page-like documents;
3. **High-frequency financial scenarios:** it emphasizes common real-world use cases such as insurance policies, reimbursement materials, and financial research reports.

FinixDocBench supports two complementary evaluation paradigms:

- **Structured parsing evaluation:** based on JSON annotations, focusing on layout structure, table structure, and reading order consistency;
- **Full-text parsing evaluation:** based on full-page Markdown representations, focusing on text fidelity and overall readability.

The annotation format is fully consistent with our training data and follows a unified 10-class tag system. Each bounding box annotation contains normalized coordinates $[x_1, y_1, x_2, y_2] \in [0, 1]^4$, a category label, and the corresponding transcribed text content. Importantly, all evaluation samples in FinixDocBench are constructed separately from the training corpus used by Data Factory and are not used in contrastive adaptation, supervised fine-tuning, or reinforcement learning.

4.1.3 Data Composition

The full FinixDocBench dataset contains approximately 5,000 pages, covering the primary document types and quality-degradation patterns encountered in real financial workflows. It is organized into four evaluation tracks: **FinixDigital**, **FinixPhoto**, **FinixHuge**, and **FinixInner**. The FinixDigital, FinixPhoto, and FinixHuge tracks are constructed around public-facing document scenarios, while FinixInner covers high-frequency internal financial workflow scenarios. To support broader research, we release a compliance-reviewed subset of FinixDocBench alongside this technical report.

The overall composition is summarized in Table 3.

FinixDocBench is organized into four complementary evaluation tracks:

- **FinixDigital:** a track for digitally native financial documents, covering financial research reports and insurance clauses (approximately 500 pages). The samples are high-definition PDFs from public-facing financial document scenarios. Characterized by dense semantic content, specialized terminology, and deep hierarchical structures, this track evaluates the model's ability to recognize domain terms, parse financial tables, and reconstruct document hierarchies accurately.

Table 3: **Data composition of FinixDocBench.**

Document Type	Source Type	Evaluation Track	Volume (Pages)
Financial Reports & Insurance Clauses	Digitally Native	FinixDigital	500
Medical Receipts	Mobile Captured	FinixPhoto	300
Large-Scale Financial Pages	Mixed Quality	FinixHuge	200
Identity Documents	Mobile Captured	FinixInner	800
Medical Examination Reports	Mobile Captured	FinixInner	800
Claim Medical Records	Mobile Captured	FinixInner	800
Expense Statements	Mobile Captured	FinixInner	800
Expense Receipts	Mobile Captured	FinixInner	800

- **FinixPhoto:** a track for camera-captured documents, containing approximately 300 pages of receipt images from the CHIP 2022 public-scenario source, all re-annotated in our unified format. Unlike the original competition annotations, our benchmark uses newly constructed full-page annotations and full-text ground truth under a unified financial document parsing schema. This subset introduces realistic low-quality factors, including motion blur, perspective distortion, and printing artifacts, and primarily evaluates model robustness under authentic mobile-capture conditions. Although prior exposure by some external models to the original CHIP 2022 source cannot be fully ruled out, these evaluation samples are not used in our own training pipeline.
- **FinixHuge:** a track for ultra-large page documents, containing approximately 200 extremely high-resolution pages with dense and complex layouts, such as ultra-long insurance clauses and very large dense tables from public-facing document scenarios. This subset primarily evaluates parsing stability and structural preservation under multi-region inference. *This subset is evaluated using the complete FinixDoc system, including the Document Preprocessing Tool Layer's split-then-merge module, rather than the standalone FinixDoc-VL model.*
- **FinixInner:** a track for high-frequency financial and insurance workflow documents, containing approximately 4,000 camera-captured pages. It covers identity documents, medical examination reports, claim medical records, expense statements, and expense receipts. These samples exhibit overlapping degradations such as shadow occlusion, blur, glare, and perspective distortion. FinixInner is used only for held-out evaluation and is never included in model training.

Across all FinixDocBench tracks, annotations are first produced as pre-annotations by our AI-driven Data Factory pipeline, and are then verified and corrected through rigorous manual quality assurance before being finalized for evaluation.

4.1.4 Evaluation Metrics

FinixDocBench deliberately excludes a standalone formula parsing metric. In real financial scenarios, formulas are sparse and rarely constitute the deployment bottleneck. Instead, the practical determinants of usability are text fidelity, table structure recovery, reading order consistency, and robustness to low-quality inputs.

Accordingly, we define the composite overall score of FinixDocBench as:

$$S_{\text{FDB}} = \frac{100(1 - E_{\text{text}}) + \text{TEDS}_{\text{table}} + 100(1 - E_{\text{order}})}{3}, \tag{45}$$

where:

- **Text Edit Distance (↓):** E_{text}, the character-level normalized edit distance computed on the full-text Markdown output [15];
- **Table TEDS / Table TEDS-S (↑):** $\text{TEDS}_{\text{table}}$ and $\text{TEDS-S}_{\text{table}}$, Tree-Edit-Distance-based Similarity scores computed on table structure predictions, where TEDS-S is invariant to the spatial order of cells [40, 43];
- **Read Order Edit Distance (↓):** E_{order}, the normalized edit distance computed on the serialized sequence of structural blocks, measuring consistency between predicted and reference reading orders [15].

In the overall score, we use $\text{TEDS}_{\text{table}}$ as the table metric, while $\text{TEDS-S}_{\text{table}}$ is reported as an auxiliary metric for order-invariant table evaluation.

4.2 Main Results

We first validate the effectiveness of our visual adaptation strategy through an isolated ablation study. We then present comprehensive evaluations on both general and domain-specific benchmarks to demonstrate the generality, robustness, and deployment relevance of FinixDoc-VL.

4.2.1 Ablation on ViT Contrastive Learning: Driving License Recognition

To validate the effectiveness of the visual adaptation strategy introduced in Section 2.2.1, we conduct a targeted ablation study on an internal driving license recognition task. This task requires extracting key fields from suboptimal camera-captured photographs and therefore serves as a representative Low-Quality Zone setting that directly stresses the robustness of the visual encoder. The purpose of this experiment is to isolate the contribution of visual adaptation under realistic low-quality capture conditions, rather than to benchmark full financial document parsing performance.

All evaluated models are fine-tuned on approximately 20k task-specific samples and tested on a private set of roughly 4k samples. The train and test sets are fully disjoint. The evaluation metric is *Document-level Accuracy*, under which a sample is counted as correct only if every required field is extracted without error. Since this experiment is designed to isolate the effect of the visual adaptation stage, its results are reported separately and are not intended for direct comparison with the benchmark results in the subsequent sections.

Table 4: **Ablation results on the internal driving license recognition task.** Performance is measured by Document-level Accuracy.

Model	**Document-level Accuracy** (↑)
Donut	0.922
MinerU	0.924
Qwen3-VL-2B	0.917
Qwen3-VL-4B	0.929
Qwen3-VL-4B (ViT-only adapted)	**0.935**

As shown in Table 4, adapting only the visual encoder already yields a clear and consistent improvement in authentic camera-captured scenarios. Although the raw base-model ranking is not strictly monotonic with parameter scale on this task, the adapted 4B model achieves the best result overall, indicating that domain-aligned visual representation learning is more important than model size alone under this type of low-quality input. This result provides direct evidence that stronger domain-adapted visual representations are a necessary foundation for the downstream improvements in structural parsing and semantic reconstruction achieved by FinixDoc-VL.

4.2.2 General Document Parsing: OmniDocBench v1.5

We first evaluate the foundational parsing capability of FinixDoc-VL on OmniDocBench v1.5 [21]. Table 5 compares FinixDoc-VL against both specialized document parsing models [5, 20, 37, 18, 8, 9, 39] and large-scale general-purpose VLMs [24, 25, 12]. For compared baselines, we report official benchmark numbers where available; $S_{\text{ODB}}^{\text{wo-formula}}$ is recomputed from the reported text and table metrics under the same formula for shared-dimension comparison.

FinixDoc-VL demonstrates strong foundational capability on this general benchmark, achieving $S_{\text{ODB}}^{\text{wo-formula}} = 92.65$. In particular, it achieves TEDS-$S_{\text{table}} = 92.76$, together with $E_{\text{order}} = 0.055$, indicating accurate table structure recovery and strong reading-order consistency on clean, digitally native documents.

Specialized document models such as PaddleOCR-VL-1.5 and GLM-OCR remain ahead on this benchmark. This outcome is unsurprising: OmniDocBench is concentrated in the Benchmark-Converged Zone of our capability matrix, where documents are relatively clean and the task emphasis is strongly aligned with mature layout-analysis and formula-aware pipelines. Nevertheless, FinixDoc-VL substantially outperforms its initialization base Qwen3-VL-4B across all shared non-formula metrics, confirming that our training recipe preserves general parsing capability while improving the model's structural and textual fidelity.

By contrast, very large general-purpose VLMs remain noticeably weaker on structural document parsing, especially in reading order and table reconstruction. The purpose of this evaluation is therefore not to claim absolute SOTA on a general clean benchmark, but to establish that FinixDoc-VL maintains a solid general parsing foundation before moving into the more challenging financial scenarios.

Table 5: **Performance on OmniDocBench v1.5.**

Model	$S_{\text{ODB}}^{\text{wo-formula}} \uparrow$	$S_{\text{ODB}} \uparrow$	$E_{\text{text}} \downarrow$	$\text{CDM}_{\text{formula}} \uparrow$	$\text{TEDS}_{\text{table}} \uparrow$	$\text{TEDS-S}_{\text{table}} \uparrow$	$E_{\text{order}} \downarrow$
Qwen3-VL-4B	86.90	86.78	0.055	86.55	79.29	84.26	0.084
FinixDoc-VL	92.65	87.40	0.042	76.89	89.50	92.76	0.055
Specialized Models							
DeepSeek-OCR-2	90.35	89.17	0.049	86.85	85.60	90.06	0.060
Dots.OCR	90.99	88.41	0.048	83.22	86.78	90.62	0.053
MinerU 2.5	91.97	90.93	0.045	88.86	88.44	92.42	0.044
FireRed-OCR	92.61	92.07	**0.035**	90.98	88.72	92.38	0.041
Youtu-Parsing	94.45	93.37	0.042	91.22	93.10	96.47	**0.026**
GLM-OCR	**94.70**	94.35	0.045	93.65	**93.89**	**96.50**	0.047
PaddleOCR-VL-1.5	94.63	**94.50**	**0.035**	**94.21**	92.76	95.79	0.042
General-purpose VLMs							
Qwen3.5-397B-A17B	88.72	88.34	0.055	87.60	82.93	88.70	0.080
Qwen3-VL-235B-A22B-Instruct	89.66	89.15	0.069	88.14	86.21	90.55	0.068
Kimi-K2.5	90.54	89.33	0.065	86.92	87.57	91.82	0.084

4.2.3 Financial Digitally Native Documents: FinixDigital

Table 6: **Performance on FinixDigital.**

Model	$S_{\text{FDB}} \uparrow$	$E_{\text{text}} \downarrow$	$\text{TEDS}_{\text{table}} \uparrow$	$\text{TEDS-S}_{\text{table}} \uparrow$	$E_{\text{order}} \downarrow$
Qwen3-VL-4B	80.18	0.145	76.04	81.23	0.210
FinixDoc-VL	**93.19**	**0.039**	**92.07**	**93.67**	0.086
Specialized Models					
DeepSeek-OCR-2	82.80	0.139	90.00	92.32	0.277
FireRed-OCR	83.10	0.119	87.10	89.14	0.259
PaddleOCR-VL-1.5	85.41	0.116	86.12	88.26	0.183
GLM-OCR	86.28	0.121	89.44	90.99	0.185
Youtu-Parsing	89.26	0.091	87.79	90.85	0.109
Dots.OCR	90.36	0.058	89.78	92.26	0.129
MinerU 2.5	92.96	0.045	91.18	92.70	**0.078**
General-purpose VLMs					
Qwen3.5-397B-A17B	84.90	0.119	87.30	89.51	0.207
Kimi-K2.5	85.05	0.119	85.95	88.24	0.189
Qwen3-VL-235B-A22B-Instruct	87.26	0.076	82.77	85.44	0.134

As defined by the Document Parsing Capability Matrix, digitally native financial documents belong largely to the Benchmark-Converged Zone. Even in this relatively mature setting, FinixDoc-VL achieves the best S_{FDB} score of 93.19. This advantage is driven by highly accurate character-level recognition on dense financial terminology ($E_{\text{text}} = 0.039$) together with strong structural fidelity in complex financial tables ($\text{TEDS}_{\text{table}} = 92.07$).

A more important observation is that strong performance on general clean benchmarks does not directly transfer to the financial domain. For example, GLM-OCR and PaddleOCR-VL-1.5 perform extremely well on OmniDocBench, but their scores on FinixDigital drop to 86.28 and 85.41, respectively. This gap indicates that financial documents impose domain-specific requirements beyond generic layout parsing, including specialized terminology, hierarchical clauses, and complex table semantics. FinixDoc-VL closes this gap effectively by combining general VLM robustness with targeted financial-domain adaptation.

4.2.4 Camera-Captured Documents: FinixPhoto

FinixPhoto corresponds to the **Low-Quality Zone** in our capability matrix, where documents are affected by blur, perspective distortion, compression artifacts, and other authentic capture noise. As shown in Table 7, the performance landscape changes substantially relative to digitally native benchmarks.

First, specialized document models degrade sharply. Systems that perform strongly on clean digital benchmarks drop into the 40–50 range under mobile-capture conditions, indicating limited robustness

Table 7: **Performance on FinixPhoto.**

Model	$S_{\text{FDB}} \uparrow$	$E_{\text{text}} \downarrow$	$\text{TEDS}_{\text{table}} \uparrow$	$\text{TEDS-S}_{\text{table}} \uparrow$	$E_{\text{order}} \downarrow$
Qwen3-VL-4B	54.28	0.408	50.13	63.04	0.465
FinixDoc-VL	**67.03**	**0.276**	69.08	**77.69**	0.404
Specialized Models					
PaddleOCR-VL-1.5	41.28	0.512	34.54	46.72	0.595
MinerU 2.5	43.08	0.513	35.54	48.58	0.550
DeepSeek-OCR-2	43.20	0.459	30.69	43.52	0.552
GLM-OCR	45.82	0.521	50.47	60.22	0.609
FireRed-OCR	47.20	0.487	38.50	53.72	0.482
Dots.OCR	52.57	0.399	44.90	56.92	0.473
Youtu-Parsing	60.90	0.345	59.01	66.23	0.418
General-purpose VLMs					
Qwen3.5-397B-A17B	62.58	0.384	62.04	71.82	0.359
Qwen3-VL-235B-A22B-Instruct	62.65	0.359	63.55	72.13	**0.397**
Kimi-K2.5	65.55	0.325	**70.16**	77.34	0.410

once the input distribution departs from the clean document assumptions under which such models are typically trained.

Second, large general-purpose VLMs become considerably more competitive in this setting. Benefiting from broader and more diverse pretraining, models such as Kimi-K2.5 and Qwen3-VL-235B-A22B-Instruct retain substantially stronger robustness to visual corruption than most specialized document parsers.

Third, FinixDoc-VL achieves the best S_{FDB} score of 67.03 and the best $\text{TEDS-S}_{\text{table}}$ score of 77.69. Compared with its base model Qwen3-VL-4B, it improves the S_{FDB} score by 12.75 points, demonstrating that our financial-domain visual adaptation and business-aligned RL optimization effectively convert the inherent robustness of a general VLM into stronger domain-specific parsing performance. This result is especially important because it validates the central motivation of this work: in real financial deployment, robustness on low-quality documents is often more consequential than marginal gains on already converged clean benchmarks.

4.2.5 Ultra-Large Document Pages: FinixHuge

FinixHuge targets the **Underexplored Large-Scale Zone** in our Document Parsing Capability Matrix, where the dominant challenge is no longer conventional OCR difficulty alone, but the joint failure of perception, generation, and integration under extreme page scale. In this setting, the primary practical bottleneck is often not only how accurately a model parses a page, but whether the page can be processed into a valid structured output at all. Since the key issue is full-pipeline processability rather than standard single-pass page parsing alone, we adopt a usability-oriented metric suite tailored to oversized-page scenarios.

Accordingly, this subset evaluates **end-to-end system usability** rather than raw model capability under strictly controlled input settings. We compare the complete **FinixDoc** system against two representative baselines, **Qwen3-VL-235B-A22B-Instruct** and **GLM-OCR**, both of which perform direct single-pass inference on the original input image [24, 8]. By contrast, FinixDoc employs the **split-then-merge** strategy described in Section 2.1 and processes oversized images through a lightweight three-stage pipeline of *Dynamic Routing* $\rightarrow$ *Structure-Aware Segmentation* $\rightarrow$ *Reconstruction*. Specifically, inputs exceeding the effective VLM token budget are routed to the segmentation branch, partitioned into multiple structure-aware sub-regions for parallel parsing, and then merged into a page-level result through direction-aware integration followed by a final integrity check.

In addition to E_{text} ($\downarrow$), $\text{TEDS}_{\text{table}}$ / $\text{TEDS-S}_{\text{table}}$ ($\uparrow$), and E_{order} ($\downarrow$), we further report **Success Rate**. A page is counted as successful if the method returns a syntactically valid, non-empty page-level parsing result without runtime failure, severe truncation, or format errors that prevent downstream evaluation. Quality metrics are computed over successfully parsed pages, while failed pages are reflected by the Success Rate. This separation allows us to distinguish parsing quality from basic processability, which is critical in the ultra-large-page regime.

Table 8: **Performance on FinixHuge.** FinixDoc denotes the complete system, including the Document Preprocessing Tool Layer's split-then-merge (tiling-and-merging) module.

Model	Success Rate ↑	S_{FDB} ↑	E_{text} ↓	$\text{TEDS}_{\text{table}}$ ↑	$\text{TEDS-S}_{\text{table}}$ ↑	E_{order} ↓
FinixDoc	**0.92**	**68.23**	**0.357**	57.09	60.10	**0.167**
Qwen3-VL-235B-A22B-Instruct	0.68	34.85	0.847	47.05	**63.20**	0.578
GLM-OCR	0.34	38.06	0.816	**59.39**	62.43	0.636

As shown in Table 8, FinixDoc achieves a Success Rate of **0.92**, successfully parsing the vast majority of pages in this subset, while Qwen3-VL-235B-A22B-Instruct and GLM-OCR reach **0.68** and **0.34**, respectively. This gap indicates that, in the oversized-page regime, direct single-pass inference frequently fails to produce a valid parse result. In such cases, processability becomes a central bottleneck before fine-grained recognition quality can even be assessed.

On the metrics most directly tied to end-to-end usability, FinixDoc achieves the best S_{FDB} score of **68.23**, the lowest E_{text} of **0.357**, and the lowest E_{order} of **0.167**. These results suggest that the perception–generation–integration chain remains sufficiently coherent after partitioning and reconstruction, preserving strong textual fidelity and reading-order consistency.

On the two table-level metrics, $\text{TEDS}_{\text{table}}$ and $\text{TEDS-S}_{\text{table}}$, FinixDoc is competitive with the baselines but does not lead. This near-parity is itself notable, because the comparison is shaped by two confounding factors that systematically disadvantage FinixDoc. First, the baseline scores are computed only over the subset of pages that can be successfully processed by single-pass inference, which introduces a survivorship bias toward easier inputs. In contrast, FinixDoc produces valid outputs for a much larger and more difficult portion of the benchmark. Second, the split-then-merge pipeline tends to produce longer and more structurally complete table predictions. As a result, roughly **25%** of FinixDoc's table cases exceed the TEDS evaluator's **6-minute timeout threshold**; under the evaluation protocol, such timed-out cases are assigned a tree-edit distance of **1.0**, corresponding to a TEDS score of **0**. This mechanically lowers the averaged table metrics and reflects evaluator-side computational limits on long, complex predictions rather than a direct reconstruction-quality deficit.

Taken together, these results support the view that ultra-large document parsing is fundamentally a system-level problem. Simply scaling model size is insufficient once page resolution and output length exceed the effective operating range of single-pass VLM inference. The *Dynamic Routing* → *Structure-Aware Segmentation* → *Reconstruction* pipeline enables FinixDoc to substantially improve processability in the Underexplored Large-Scale Zone of our capability matrix.

4.2.6 Internal Financial Domain Evaluation: FinixInner

FinixInner is an internal held-out evaluation track containing 4,000 camera-captured documents from high-frequency financial and insurance workflows. Following the same evaluation protocol as FinixDocBench, we report the **Overall** score for each document category, scaled to the range $[0, 100]$ for consistency with the preceding benchmark results.

Table 9: **Performance on FinixInner.**

Model	Overall	Claim Medical Records	Expense Statements	Expense Receipts	Medical Examination Reports	Identity Documents
Qwen3-VL-4B	66.42	50.24	64.85	70.55	60.21	86.26
FinixDoc-VL	**84.08**	**85.42**	**80.79**	85.15	**78.38**	90.64
Specialized Models						
DeepSeek-OCR-2	52.77	42.40	54.37	66.92	53.64	46.51
FireRed-OCR	64.93	50.23	62.67	66.24	69.45	76.08
PaddleOCR-VL-1.5	61.06	45.17	65.19	62.86	56.30	75.79
MinerU 2.5	62.53	55.85	62.76	65.28	73.35	55.41
GLM-OCR	70.26	62.45	62.68	58.25	76.43	91.51
Dots.OCR	72.54	64.36	71.44	73.16	72.99	80.77
Youtu-Parsing	78.73	62.81	74.55	**88.08**	75.31	92.90
General-purpose VLMs						
Qwen3.5-397B-A17B	70.83	58.24	73.61	75.33	61.22	85.77
Kimi-K2.5	76.91	76.32	73.35	80.75	66.71	87.41
Qwen3-VL-235B-A22B-Instruct	76.32	60.08	79.27	80.86	68.30	**93.10**

Table 9 shows that FinixDoc-VL delivers the strongest and most balanced performance across the FinixInner benchmark. In particular, it achieves the best results on the three most challenging subsets: *Claim Medical Records* (85.42), *Expense Statements* (80.79), and *Medical Examination Reports* (78.38). These categories are characterized by dense domain terminology, complex table structures, non-trivial reading orders, and severe quality degradation caused by mobile capture. The strong gains on these subsets indicate that FinixDoc-VL is especially effective in scenarios where visual robustness and structural consistency must be maintained simultaneously.

Several observations are noteworthy. First, specialized document models exhibit substantial variance across document categories. While some remain highly competitive on standardized formats—for example, Youtu-Parsing achieves the best result on *Expense Receipts* (88.08) and also performs strongly on *Identity Documents* (92.90)—their performance deteriorates noticeably on more heterogeneous and structurally complex categories such as *Claim Medical Records*. This pattern suggests that strong results on standardized or cleaner layouts do not necessarily transfer to real-world financial documents with compounded visual and structural difficulties.

Second, very large general-purpose VLMs are competitive on template-driven categories. In particular, Qwen3-VL-235B-A22B-Instruct achieves the best score on *Identity Documents* (93.10), likely benefiting from stronger general visual robustness and broader template priors acquired during large-scale pretraining. However, its score on *Claim Medical Records* drops to 60.08, indicating that general-purpose capability alone is insufficient for reliably handling dense, noisy, and structurally demanding financial documents.

Overall, the FinixInner results reinforce our central claim: progress in industrial financial document parsing should not be judged solely by performance on clean or widely used benchmarks. Instead, it should be measured by whether a system can robustly process low-quality, high-complexity documents encountered in real deployment environments. Under this criterion, FinixDoc-VL demonstrates the strongest overall robustness and the best practical readiness among the evaluated models.

5 Related Work

5.1 Traditional OCR Methods and Document Parsing Pipelines

Traditional document parsing systems typically adopt a modular pipeline architecture, decomposing the overall task into multiple stages such as optical character recognition (OCR), layout analysis, table structure parsing, key information extraction, and rule-based post-processing [32, 41, 4]. These methods have high engineering maturity and are well suited to scenarios with stable templates, regular layouts, and clearly defined task boundaries. As a result, they have long formed the backbone of industrial document digitization systems, including invoice processing, form understanding, and document archiving. OCR systems such as PaddleOCR and the toolchains built around them remain core components of many production systems today [4].

However, pipeline-based methods also suffer from several well-known limitations. First, different submodules usually require separate design, training, and hyperparameter tuning, resulting in substantial system integration complexity and high maintenance costs. Second, errors from upstream OCR or layout detection can propagate and amplify in later stages such as structure recovery and field extraction, thereby constraining the end-to-end performance ceiling. Third, these methods often rely heavily on domain-specific rules, template assumptions, and human heuristics, which limits their transferability across templates and application scenarios. With the development of end-to-end document understanding methods, an increasing number of studies have begun to reduce their dependence on explicit OCR pipelines, instead reformulating document parsing as direct generation from page images to structured outputs [11, 14, 2]. Representative works such as Donut demonstrated that OCR-free end-to-end document understanding is feasible, laying the groundwork for subsequent unified generative document parsing methods [11].

5.2 General Vision-Language Models for Document Understanding

In recent years, the rapid development of general vision-language models (VLMs) has significantly advanced document understanding [1, 24, 34]. Unlike traditional pipelines that separately address OCR, structure recovery, and semantic extraction, general VLMs tend to jointly model visual appearance, textual content, spatial layout, and cross-modal semantic relationships within a unified framework. This enables them to support a wide range of tasks, including page-level understanding, document question answering, structured extraction, and content generation. Representative model families include Qwen-VL and Kimi-VL, which have shown strong generalization and cross-task transfer ability across various document understanding tasks [1, 24, 13].

Compared with traditional OCR pipelines, a key advantage of general VLMs is their unified modeling capability: they can directly generate structured outputs from raw document images, simplifying system design and providing a more direct interface for downstream reasoning tasks [11, 14, 1]. This is particularly useful in scenarios where parsed document content must be passed to large language models (LLMs) for question answering, summarization, or decision support. More importantly, due to their larger scale, broader pretraining coverage, and stronger open-world visual priors, general VLMs often exhibit better robustness than document-specific models under distribution shifts such as blur, shad-

ows, perspective distortion, and other image degradations. However, because they are not explicitly optimized for document parsing, they may still exhibit insufficient accuracy on dense tables, complex layouts, fine-grained character recognition, and long structured outputs. When visual evidence is insufficient, they may also rely excessively on language priors and generate plausible but ungrounded outputs, leading to hallucinations [17, 19]. In high-stakes financial document parsing scenarios, such behavior is often unacceptable.

5.3 Specialized Models for Document Parsing and OCR

In addition to general VLMs, a growing number of models specifically optimized for document parsing and OCR have emerged [5, 20, 37, 18, 8]. These methods are typically designed around document-specific challenges, including dense text recognition, layout structure recovery, reading order reconstruction, table parsing, and structured result generation. Representative models include MinerU 2.5, Dots.OCR, GLM-OCR, DeepSeek-OCR-2, and PaddleOCR-VL-1.5. Compared with general multimodal models, these specialized systems usually incorporate stronger document priors and place greater emphasis on structural fidelity, page-level reconstruction quality, and parsing stability.

On mainstream benchmarks, specialized document models usually perform strongly on high-quality scans, digitally native documents, and relatively simple page layouts [21, 5, 20, 18, 8]. They often surpass general-purpose models in tasks such as table parsing, reading order recovery, and preserving structural consistency. However, these gains are largely concentrated on clean and regular document distributions. Many existing methods are trained and evaluated mainly on digitally native pages, scanned copies, or synthetic data, while camera-captured documents—common in real-world business workflows—remain underrepresented. As a result, even though specialized models have achieved impressive benchmark performance, their robustness, stability, and scalability in real-world financial settings, especially in low-quality and large-scale scenarios, still require further improvement.

Overall, existing work has made substantial progress on clean and moderately complex document parsing, but the capability boundary under low-quality, large-scale, and risk-sensitive financial scenarios remains insufficiently explored. This gap motivates both the capability matrix and the system design proposed in this work.

6 Conclusion

In this report, we study the key challenges of document parsing in real-world financial scenarios. We analyze why current methods can achieve strong benchmark performance yet still fall short in practical deployment, especially when documents are low-quality, structurally complex, or extremely large. To make this gap explicit from an industrial perspective, we introduce a *Document Parsing Capability Matrix*. To address it in practice, we present **FinixDoc**, an end-to-end agentic document parsing system tailored to the requirements of financial applications, together with a domain-adapted training recipe, an AI-driven human-in-the-loop Data Factory pipeline, and a benchmark designed for realistic financial deployment settings.

We argue that progress in financial document parsing should not be measured solely by performance on conventional benchmarks concentrated in the Benchmark-Converged Zone. Instead, evaluation should prioritize stability, robustness, and usability when models face low-quality, high-complexity, and large-scale documents. From this perspective, FinixDoc not only provides a practical parsing system for financial applications, but also highlights the need to rethink how document parsing systems are evaluated and improved in industrial settings.

Several promising directions remain for future work. First, in the Low-Quality and Ambiguous-Unrecoverable Zones, developing reliable generation strategies based on uncertainty estimation and refusal mechanisms will be crucial for financial scenarios, where missing a field is often preferable to extracting an incorrect one. Second, progress in the Underexplored Large-Scale Zone will likely require a more unified multimodal architecture that can jointly model fine-grained local perception, global document structure, and consistency across distant regions. Finally, we will continue expanding FinixDocBench to cover a broader range of real-world financial sub-scenarios and more fine-grained task settings, so as to encourage more systematic benchmarking of industrial document parsing capabilities.

Taken together, the capability matrix, the FinixDoc system, the associated training recipe, the human-in-the-loop Data Factory pipeline, and the evaluation benchmarks introduced in this work provide a practical foundation for advancing financial document parsing under real deployment constraints. More broadly, we hope this work can help shift document parsing research beyond already converged clean benchmarks and toward more robust, verifiable, and deployment-oriented capabilities.

References

[1] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[2] Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents. *arXiv preprint arXiv:2308.13418*, 2023.

[3] 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 0.9b ultra-compact vision-language model. *arXiv preprint arXiv:2510.14528*, 2025.

[4] Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, Yue Zhang, Wenyu Lv, Kui Huang, Yichao Zhang, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. PaddleOCR 3.0 technical report. *arXiv preprint arXiv:2507.05595*, 2025.

[5] Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, Yue Zhang, Yubo Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. PaddleOCR-VL-1.5: Towards a multi-task 0.9b VLM for robust in-the-wild document parsing. *arXiv preprint arXiv:2601.21957*, 2026.

[6] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[8] 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.

[9] FireRed Team. FireRed-OCR technical report. *arXiv preprint arXiv:2603.01840*, 2026.

[10] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18661–18673, 2020.

[11] Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. OCR-free document understanding transformer. In *European Conference on Computer Vision*, pp. 498–517. Springer, 2022.

[12] Kimi Team. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

[13] Kimi Team, Angang Du, Bohong Yin, Bowei Xing, Bowen Qu, Bowen Wang, et al. Kimi-VL technical report. *arXiv preprint arXiv:2504.07491*, 2025.

[14] Kenton Lee, Mandar Joshi, Iulia Raluca Turc, Hexiang Hu, Fangyu Liu, Julian Martin Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. Pix2Struct: Screenshot parsing as pretraining for visual language understanding. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 18893–18912, 2023.

[15] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[16] Minghao Li, Yiheng Xu, Lei Cui, Shaohan Huang, Furu Wei, Zhoujun Li, and Ming Zhou. DocBank: A benchmark dataset for document layout analysis. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 949–960, 2020.

[17] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 292–305, 2023.

[18] 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.

[19] Hanchao Liu, Wenyuan Xue, Yifei Chen, Dapeng Chen, Xiutian Zhao, Ke Wang, Liping Hou, Rongjun Li, and Wei Peng. A survey on hallucination in large vision-language models. *arXiv preprint arXiv:2402.00253*, 2024.

[20] 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. *arXiv preprint arXiv:2509.22186*, 2025.

[21] Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. OmniDocBench: Benchmarking diverse PDF document parsing with comprehensive annotations. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24838–24848, 2025.

[22] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.

[23] Birgit Pfitzmann, Christoph Auer, Michele Dolfi, Ahmed S. Nassar, and Peter W. J. Staar. DocLayNet: A large human-annotated dataset for document-layout segmentation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3743–3751, 2022.

[24] Qwen Team. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[25] Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, 2026. Official release blog.

[26] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 8748–8763, 2021.

[27] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[28] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023.

[29] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, 2015.

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[31] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[32] Ray Smith. An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition*, volume 2, pp. 629–633. IEEE, 2007.

[33] Ting Sun, Cheng Cui, Yuning Du, and Yi Liu. PP-DocLayout: A unified document layout detection model to accelerate large-scale data construction. *arXiv preprint arXiv:2503.17213*, 2025.

[34] V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, Shuaiqi Duan, et al. GLM-4.5V and GLM-4.1V-Thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv preprint arXiv:2507.01006*, 2025.

[35] Baode Wang, Biao Wu, Weizhen Li, Meng Fang, Zuming Huang, Jun Huang, Haozhe Wang, Yanjie Liang, Ling Chen, Wei Chu, and Yuan Qi. Infinity Parser: Layout aware reinforcement learning for scanned document parsing. *arXiv preprint arXiv:2506.03197*, 2025.

[36] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[37] Haoran Wei, Yaofeng Sun, and Yukun Li. DeepSeek-OCR 2: Visual causal flow. *arXiv preprint arXiv:2601.20552*, 2026.

[38] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[39] Youtu-Parsing Team. Youtu-Parsing: Perception, structuring and recognition via high-parallelism decoding. *arXiv preprint arXiv:2601.20430*, 2026.

[40] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.

[41] Qintong Zhang, Bin Wang, Victor Shea-Jay Huang, Junyuan Zhang, Zhengren Wang, Hao Liang, Conghui He, and Wentao Zhang. Document parsing unveiled: Techniques, challenges, and prospects for structured information extraction. *arXiv preprint arXiv:2410.21169*, 2024.

[42] Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. PubLayNet: Largest dataset ever for document layout analysis. In *2019 International Conference on Document Analysis and Recognition*, pp. 1015–1022. IEEE, 2019.

[43] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. Image-based table recognition: Data, model, and evaluation. In *European Conference on Computer Vision*, pp. 564–580. Springer, 2020.

A Implementation Details of the Split-then-Merge Strategy

As discussed in Section 2.1, the **Document Preprocessing Tool Layer** in FinixDoc includes a lightweight **split-then-merge** strategy for handling the *Underexplored Large-Scale Zone* of the Document Parsing Capability Matrix, namely ultra-large-page documents that are difficult to process reliably through a single native VLM call. This strategy is not intended to replace the core parsing capability of FinixDoc-VL. Instead, it serves as an external engineering supplement that improves the tractability and stability of oversized-image parsing in practical deployment.

The design objective of this strategy is to transform ultra-large-page parsing from a single monolithic inference step into a modular pipeline that supports **adaptive routing**, **local-region parsing**, and **page-level result integration**. In practice, this design allows the system to preserve the native direct-parsing path for standard pages, while activating a structured split-then-merge process only when page scale or layout complexity makes single-pass inference unreliable.

As illustrated in Figure 8, the strategy follows a three-stage internal pipeline: *Dynamic Routing* → *Structure-Aware Segmentation* → *Reconstruction*. Oversized images are first analyzed by a lightweight routing stage, then either sent directly to FinixDoc-VL or partitioned into multiple local regions for independent parsing, and finally merged into a unified page-level output.

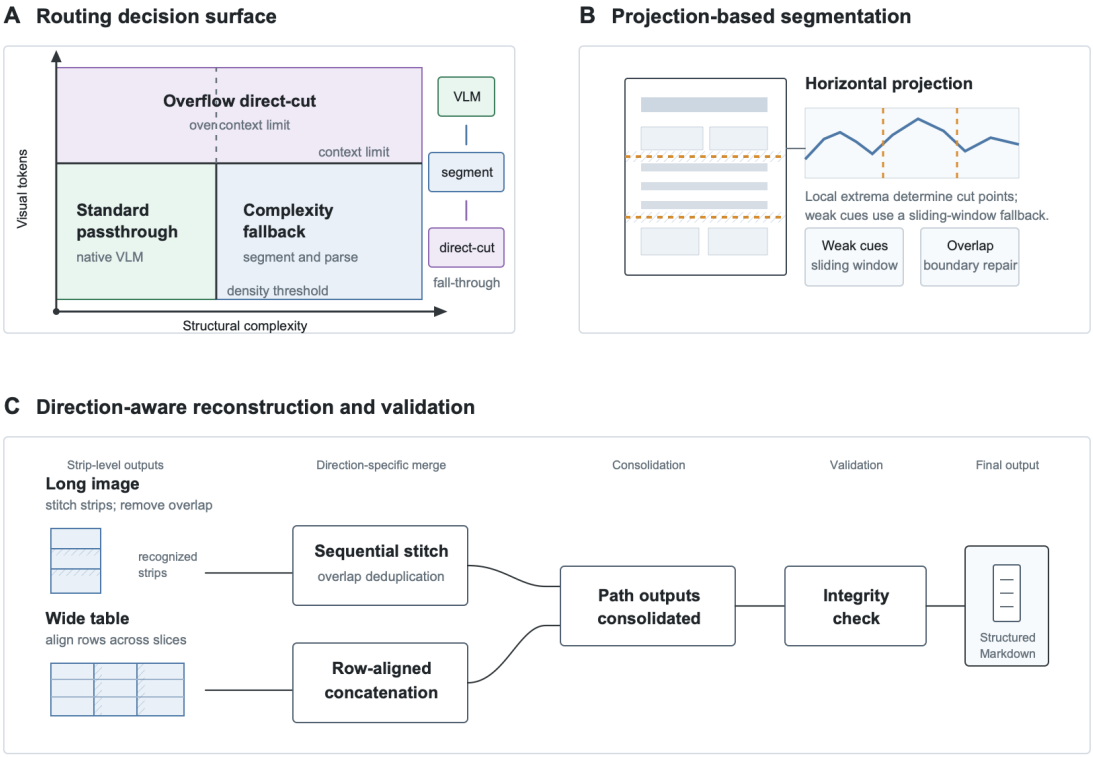

Figure 8: **Internal pipeline of the split-then-merge strategy in the Document Preprocessing Tool Layer.** The strategy follows a three-stage process: Dynamic Routing, Structure-Aware Segmentation, and Reconstruction. Depending on token feasibility and structural complexity, an input page is processed either by direct VLM parsing or by a split-then-merge path for ultra-large-page handling.

A.1 Dynamic Routing for Ultra-Large Pages

Given an input image, the strategy first extracts a set of coarse page-level features, including pixel scale, aspect ratio, and estimated information density. Based on these signals, the page is dispatched in the two-dimensional space of *Pixel Scale* × *Structural Complexity* through a three-way routing strategy:

- **Overflow-to-Splitting.** If the estimated visual token count exceeds the effective input budget of the underlying VLM, the page is routed directly to the split branch.
- **Direct Parsing.** If the token count remains within the feasible range and the page layout is relatively regular, the image is processed through the native FinixDoc-VL path and the result is

returned directly.

- **Complexity Fallback.** If the token count is nominally acceptable but the page exhibits excessive semantic density or structural complexity—for example, dense large tables, ultra-long clauses, or mixed text-table layouts—the strategy falls back to the split branch to avoid unstable single-pass parsing.

In our implementation, the token-load estimate is derived from the effective visual patch count under the native visual tokenizer, while the structural complexity score is computed using lightweight heuristics based on page geometry, projection statistics, and coarse content-density indicators. This routing strategy preserves efficiency on ordinary pages while providing a fail-safe path for difficult oversized inputs. Rather than forcing all pages through segmented inference, the system activates splitting only when necessary.

A.2 Structure-Aware Splitting Strategy

For pages routed to the split branch, the strategy does not simply apply uniform equal-size partitioning. Instead, it uses a **structure-aware segmentation strategy** based on binarized orthogonal projection, with the goal of choosing cut locations that are better aligned with document layout.

This strategy includes three key mechanisms:

- **Structure-Aligned Cutting.** According to the geometric profile of the page, the strategy computes horizontal or vertical projection maps and selects cut points near local extrema. After binarization, foreground density is accumulated along the target axis to form the projection profile used for cut-point selection. In practice, these locations tend to correspond to whitespace bands or low-density regions, which reduces the probability of splitting text lines, clause boundaries, or table rows across split regions.
- **Weak-Structure Fallback.** When projection signals are weak or ambiguous, the strategy falls back to a sliding-window splitting scheme. In this mode, cut positions are still chosen using projection-aware heuristics, so that the method remains applicable even on pages without strong regular layout cues.
- **Split Overlap.** Adjacent split regions preserve a fixed overlap region. This overlap provides local boundary context for subsequent deduplication, boundary repair, and page-level integration, especially when long text spans or table structures cross split boundaries.

Through this design, ultra-large pages are decomposed into a set of local regions that better preserve reading continuity and structural integrity than naive uniform splitting, while remaining suitable for independent parsing by FinixDoc-VL and amenable to parallel execution in deployment.

A.3 Reconstruction and Merging

After all local regions are parsed independently, the strategy performs page-level reconstruction through **direction-aware merging**. The merging strategy is selected according to the dominant geometric orientation of the page:

- **Vertical merging for ultra-long pages.** For long-form pages, local outputs are stitched sequentially along the vertical direction. Overlap-aware deduplication is used to suppress repeated fragments and repair boundary inconsistencies introduced during splitting.
- **Horizontal merging for ultra-wide layouts.** For extremely wide pages, especially dense large tables, local outputs are integrated by row-aware horizontal concatenation, using overlap regions and line-level alignment cues to preserve row correspondence as much as possible and improve cell continuity across adjacent regions.

The merged result is then passed through a final integrity-check stage, which verifies global consistency, suppresses duplicated content, repairs minor structural discontinuities, and ensures that the final page-level output conforms to the target Markdown representation used throughout FinixDoc. This stage is particularly important for preserving content completeness and reading-order consistency on FinixHuge-style pages.

A.4 End-to-End Control Flow

For clarity, we summarize the full inference logic of the split-then-merge strategy in Algorithm 1.

Algorithm 1 Split-then-Merge Pipeline for Ultra-Large Pages

Require: Input image I, base parser $\mathcal{M}$, token budget τ
Ensure: Structured Markdown output Y
Extract page-level features $f(I)$, including pixel scale, aspect ratio, and information density
Estimate effective token load $\hat{t}(I)$ and structural complexity score $\hat{c}(I)$
if $\hat{t}(I) > \tau$ **then**
    Route I to the split branch
else if $\hat{t}(I) \leq \tau$ **and** $\hat{c}(I)$ is low **then**
    $Y \leftarrow \mathcal{M}(I)$ ▷ Direct FinixDoc-VL parsing
    return Y
else
    Route I to the split branch ▷ Complexity fallback
end if
Determine the primary splitting direction based on page geometry
Compute orthogonal projection profiles along the target axis
Select structure-aware cut points; if unstable, switch to sliding-window fallback
Generate overlapping local regions $\{I_k\}_{k=1}^{K}$
for all local regions I_k **do**
    $Y_k \leftarrow \mathcal{M}(I_k)$
end for
if the primary splitting direction is vertical **then**
    Merge $\{Y_k\}$ by vertical stitching with overlap deduplication
else
    Merge $\{Y_k\}$ by row-aware horizontal concatenation
end if
Perform final integrity checking and lightweight structural repair
return final Markdown output Y

A.5 Discussion

The split-then-merge strategy does not alter the core model architecture of FinixDoc-VL. Instead, it provides a lightweight engineering mechanism within the **Document Preprocessing Tool Layer** that makes ultra-large-page parsing more tractable in deployment. For standard pages, the direct-parsing path is preserved, with only minimal routing overhead before native FinixDoc-VL parsing. For oversized pages, the split branch reduces context-length pressure and memory pressure by converting a single difficult page into multiple manageable local regions. The subsequent reconstruction stage then restores page-level continuity as much as possible.

Overall, this strategy should be viewed as a practical engineering extension of FinixDoc for the *Under-explored Large-Scale Zone*, rather than as a standalone parsing model. Its role is to improve usability and stability on ultra-large inputs by combining adaptive routing, structure-aware splitting, and direction-aware merging within the Document Preprocessing Tool Layer.

B Financial Homoglyph Vocabulary and Auxiliary Lexicons

To support the hard negative construction process described in Section 2.2.1, we build a **financial homoglyph vocabulary** that stores visually confusable characters frequently observed in real financial documents. The construction of this vocabulary relies on multi-view similarity filtering, which is supported by two foundational dictionaries: a stroke-order lexicon and a stroke-count lexicon [36, 15, 27]. This section details the resulting vocabulary and its underlying auxiliary lexicons.

B.1 Financial Homoglyph Vocabulary

We use the term *homoglyph* in a broad practical sense to include both strict glyph-level lookalikes and near-homoglyph confusions that become difficult to distinguish in low-quality document images. This broader definition is particularly relevant in financial deployment, where visually small deviations in digits, field values, names, and medical or reimbursement terminology may lead to disproportionately large downstream business risk.

The vocabulary covers multiple character types, including Arabic numerals, uppercase and lowercase English letters, Roman numerals, symbol-like characters, and high-frequency Chinese characters commonly appearing in financial, insurance, medical, and reimbursement documents. Compared with generic visually similar character lists, this vocabulary is specifically tailored to the financial domain and emphasizes confusions that are both visually plausible and practically risky in downstream business scenarios.

Formally, for a character c, the vocabulary provides a candidate set

$$\mathcal{H}(c) = \{h_1, h_2, \ldots, h_m\},$$

where each h_i denotes a visually similar or perceptually confusable alternative that may replace c during hard negative construction. These candidate sets are used to generate negative text samples by selectively substituting characters in the ground-truth annotation with their corresponding candidates. This process yields hard negatives that remain lexically close to the positive target while introducing realistic visual ambiguity.

The full vocabulary is constructed through candidate retrieval followed by similarity-based filtering; the entries below are illustrative rather than exhaustive. Table 10 shows representative examples from the vocabulary. In implementation, the vocabulary is stored as a key–value mapping from target characters to candidate homoglyph sets. A simplified example is shown below:

```
{
   "0": ["O", "8", "6", "9", "D"],
   "I": ["l", "1", "|", "!"],
   "|": ["I", "1", "!", "｜"],
   "维": ["唯", "淮", "推", "准"],
   "清": ["靖", "情", "请", "洁"],
   "痛": ["疼", "病", "痹", "痪"]
}
```

During training data construction, we sample replacements from this vocabulary under predefined rules so that the generated negative samples better match real OCR and parsing errors observed in financial deployment. In this way, the contrastive learning stage is encouraged to distinguish not only semantically different strings, but also visually confusable alternatives that are highly relevant to practical business risk.

Table 10: Representative examples from the financial homoglyph vocabulary.

Target Character	Example Homoglyphs	Target Character	Example Homoglyphs
0	O, 8, 6, 9, D	维	唯, 淮, 推, 准
O	0, Q, D	清	靖, 情, 请, 洁
I	l, 1, |, !	请	情, 清, 谱, 讲, 证
l	I, i, |, !	组	细, 织, 绪, 级, 纸
|	I, l, !, |	痛	疼, 病, 痹, 痪
1	I, l, 7, !	靖	情, 晴, 清
B	8, D, P	情	请, 清, 青, 倩
S	5, 8	细	维, 组, 纸, 绪
C	G, 0, 6	浩	洁, 活, 清
D	O, 0, B	继	维, 续, 编, 细
T	I, 7	准	淮, 推, 难, 维
U	V, O, D	唯	维, 淮, 谁

B.2 Stroke-Order Lexicon

To compute the stroke-order similarity $S_{\text{order}}(c_i, c_j)$ during candidate retrieval, we construct a stroke-order lexicon covering over 20,000 common Chinese characters. This lexicon is essential for capturing the internal composition of characters that cannot be fully resolved by image-level structural similarity (SSIM) alone [36].

As shown in the snippet below, each character is decomposed into two aligned lists: `stroke_type_sequence`, which records the normalized geometric type of each stroke using a compact stroke-code nomenclature (e.g., `S` for Shu/Vertical, `H` for Heng/Horizontal, `P` for Pie/Left-Falling, `D` for Dian/Dot, and `Z` for Zhe/-Turning strokes, including complex multi-turn variants), and `stroke_name_sequence`, which provides the corresponding descriptive stroke name. During candidate retrieval, these sequences allow us to robustly estimate character similarity using Dynamic Time Warping (DTW) [27] and Longest Common Subsequence (LCS), which are resilient to localized order shifts. Additionally, alternative stroke names (e.g., separated by slashes in `stroke_name_sequence`) are included to accommodate variant naming conventions and enhance matching robustness.

A simplified snippet of the stroke-order data structure is provided below:

```
{
   "𠂇": {
      "stroke_type_sequence": [
```

```
            "S", "Z", "H", "D", "P", "S"
        ],
        "stroke_name_sequence": [
            "竖", "横折", "横", "点", "撇", "竖"
        ]
    },
    "阿": {
        "stroke_type_sequence": [
            "Z_multi", "S", "H", "S",
            "Z", "H", "S_hook"
        ],
        "stroke_name_sequence": [
            "横折折折钩/横撇弯钩", "竖", "横", "竖",
            "横折", "横", "竖钩"
        ]
    },
    "啊": {
        "stroke_type_sequence": [
            "S", "Z", "H", "Z_multi", "S",
            "H", "S", "Z", "H", "S_hook"
        ],
        "stroke_name_sequence": [
            "竖", "横折", "横", "横折折折钩/横撇弯钩",
            "竖", "横", "竖", "横折", "横", "竖钩"
        ]
    }
}
```


B.3 Stroke-Count Lexicon

The stroke-count lexicon provides a lightweight auxiliary cue for writing complexity, which directly supports the computation of the stroke-count similarity $S_{\text{count}}(c_i, c_j)$. In practical financial document parsing, characters with identical or highly similar stroke counts (e.g., "清" and "情") are particularly susceptible to confusion under severe image degradation, such as camera blur or low optical resolution.

This lexicon maps each candidate character directly to its integer stroke count. To illustrate the distribution and coverage of this lexicon, Table 11 presents representative characters grouped by their respective stroke counts.

Table 11: Representative examples from the stroke-count lexicon.

Stroke Count	Example Characters
1	一, 丨, 丶, 丿, 乙
2	丁, 七, 乃, 九, 了, 二, 乜
3	万, 丈, 三, 上, 下, 久, 义, 亡, 个, 丫
4	不, 丑, 专, 中, 丰, 乌, 书, 云, 互, 五
5	且, 世, 丘, 业, 东, 丝, 乎, 乏, 乐, 主
6	丞, 丢, 两, 争, 亘, 亚, 交, 亥, 亦, 产

C Qualitative Comparison Results on FinixDigital and FinixPhoto

This appendix provides qualitative comparison results of **FinixDoc** against the best specialized model and the best general-purpose VLM selected for each of the **FinixDigital** and **FinixPhoto** evaluation tracks. For each subset, we select representative samples that highlight the typical structural, lexical, and robustness challenges of real-world financial document parsing.

The comparison is organized along two evaluation axes that match the design of the two tracks:

- **FinixDigital** (financial digitally native documents): we compare against **MinerU 2.5** as the best specialized model and **Qwen3-VL-235B-A22B-Instruct** as the best general-purpose VLM. The selected samples emphasize dense table structures, deep heading hierarchies, and domain-specific formatting such as superscript footnote references.
- **FinixPhoto** (camera-captured documents): we compare against **Youtu-Parsing** as the best specialized model and **Kimi-K2.5** as the best general-purpose VLM. The selected samples emphasize OCR-level robustness under motion blur, perspective distortion, and printing artifacts, as well as the tendency of competing models to hallucinate content beyond the visible image.

As illustrated in Figures 9–14, each example uses a four-way side-by-side layout: the original page (or per-field overlay), the FinixDoc output, the best specialized model output, and the best general-purpose model output. The primary purpose of this layout is to enable a direct comparison of structural fidelity, content accuracy, and robustness across different systems under the same input conditions.

C.1 FinixDigital: Comparison with MinerU 2.5 and Qwen3-VL-235B-A22B-Instruct

Figures 9–11 showcase FinixDoc's performance on digitally native financial documents from the FinixDigital track. Each figure presents a side-by-side comparison of the rendered Markdown output: the original page image, FinixDoc's parsed and rendered result, the result of the best specialized model (MinerU 2.5), and the result of the best general-purpose VLM (Qwen3-VL-235B-A22B-Instruct).

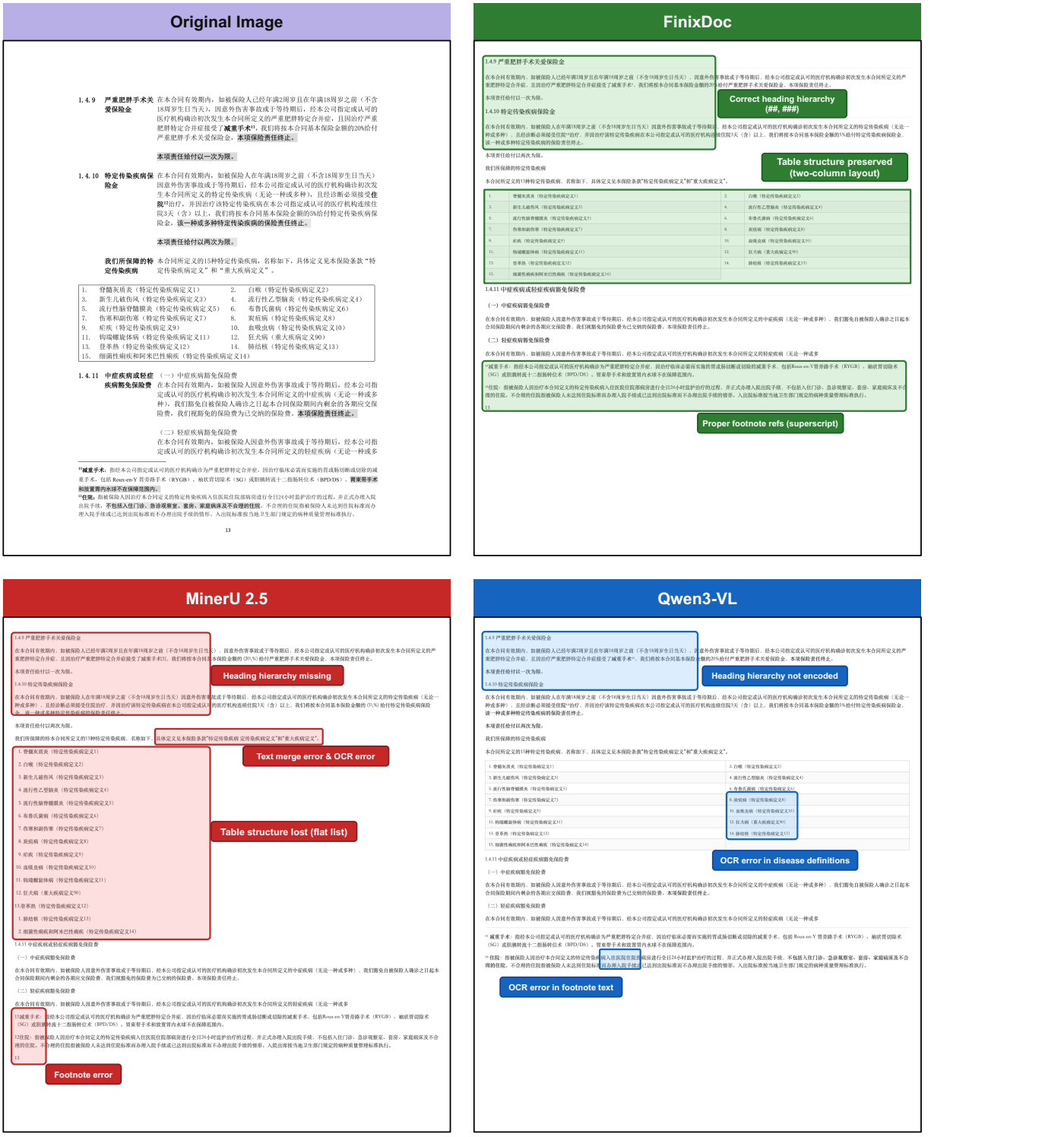

Figure 9: **FinixDigital Sample 1.** Comparison of parsing results for an insurance clause page containing a two-column disease list table and footnote references. FinixDoc correctly preserves the two-column table structure for the 15 infectious diseases, maintains proper heading hierarchy (##, ###), and retains superscript footnote references ([11], [12]). In contrast, MinerU 2.5 collapses the table into a flat list, fails to encode the heading hierarchy, merges neighboring text around the disease-list introduction, and introduces OCR errors. Qwen3-VL-235B-A22B-Instruct preserves the table layout more completely than MinerU 2.5, but still fails to encode the heading hierarchy, changes domain terms in the disease-list entries (e.g., 特定传染疾病定义 becomes 特定传染病定义), and introduces a footnote-level OCR error (e.g., 入住医院 becomes 入往医院).

Figure 10: **FinixDigital Sample 2.** Comparison of parsing results for an insurance product specification page containing a financial illustration table and section headings. FinixDoc preserves the numeric table grid, reconstructs the heading hierarchy (`### 3`, `#### 3.1`, `#### 3.2`), and maintains consistent list formatting with (1) and (2) numbering. MinerU 2.5 fails to render the circled section marker and subsection hierarchy as Markdown headings and mixes bullet styles in the notes. Qwen3-VL-235B-A22B-Instruct misinterprets the first numeric data row as a table header and introduces an HTML/dataframe-style table with an extra index-like column.

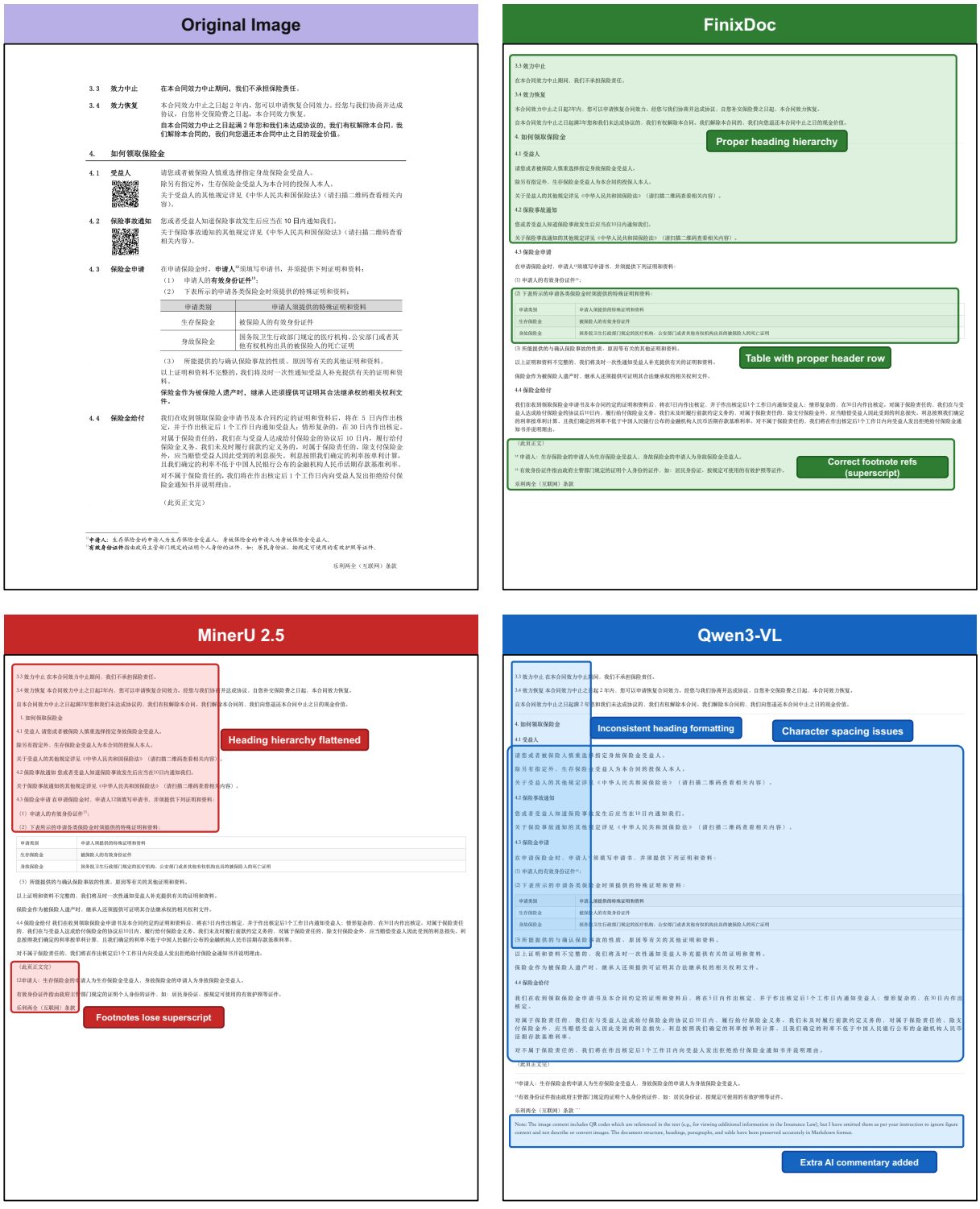

Figure 11: **FinixDigital Sample 3.** Comparison of parsing results for an insurance contract page with nested headings, a structured table, and footnote references. FinixDoc accurately reconstructs the heading hierarchy (`### 3.3`, `### 3.4`, `## 4`, `### 4.1`–`### 4.4`), preserves the table with proper headers, and retains superscript footnote references ([12], [13]). MinerU 2.5 flattens the heading hierarchy, loses or inconsistently renders superscript footnotes (e.g., [12] becomes plain text), and degrades the page-ending footnote area. Qwen3-VL-235B-A22B-Instruct preserves the table but inserts spurious spacing between Chinese characters, appends extraneous AI commentary after the parsed page, and shows inconsistent heading formatting.

C.2 FinixPhoto: Comparison with Youtu-Parsing and Kimi-K2.5

Figures 12–14 illustrate FinixDoc's robustness on camera-captured document images from the FinixPhoto track. Each figure is organized as a 2×2 grid composed of (i) the unannotated original image, and (ii)–(iv) the same original image overlaid with the ground-truth bounding boxes of selected fields, color-coded by whether each model parsed that field correctly (green) or incorrectly (red). For incorrect fields, the model's actual output is rendered next to the corresponding box so that the reader can see the exact error mode. All bounding boxes are taken from the FinixPhoto ground-truth annotations.

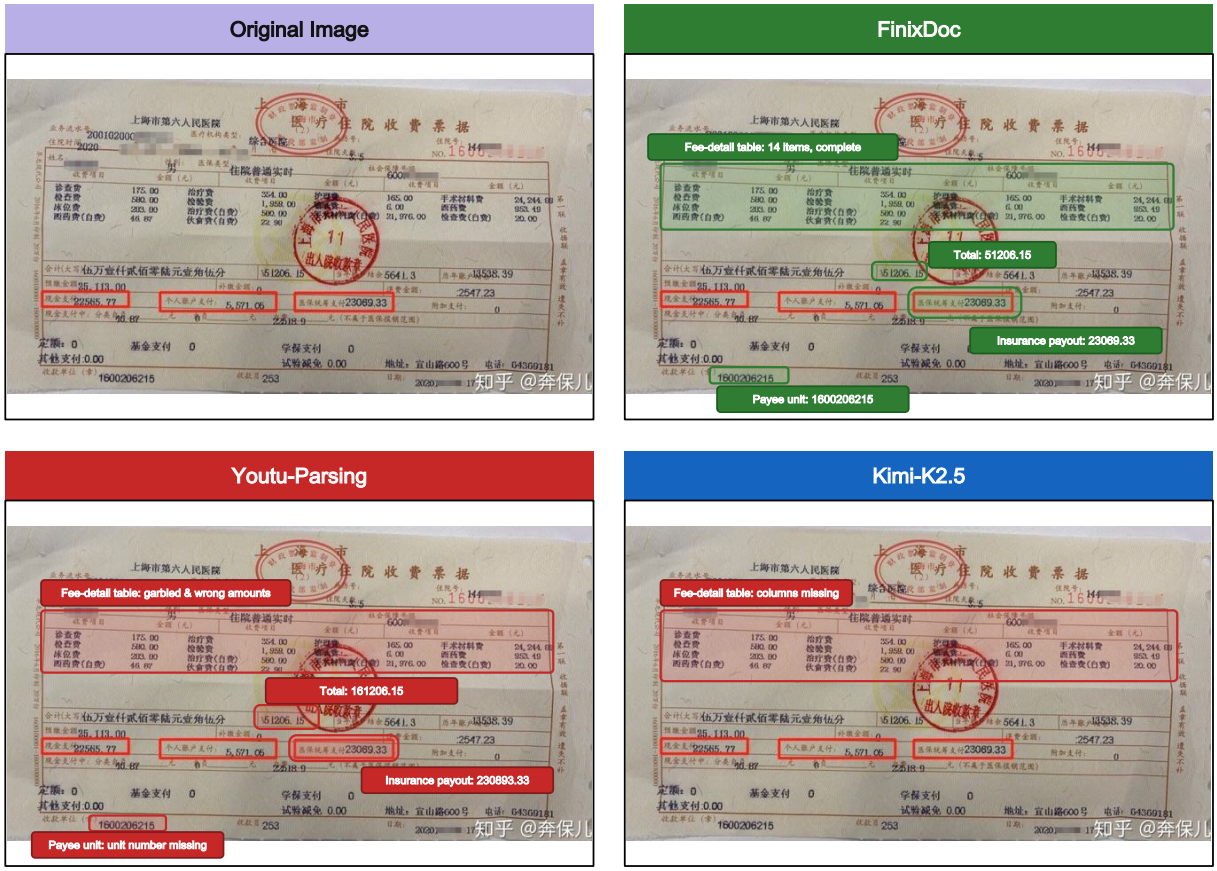

Figure 12: **FinixPhoto Sample 1.** Per-field comparison on a Shanghai inpatient receipt captured under challenging conditions. Four fields are highlighted: the fee-detail table, the total amount (`51206.15`), the medical-insurance payout (`23069.33`), and the payee-unit number (`1600206215`). FinixDoc parses all four correctly. Youtu-Parsing reads the total as `161206.15`, the insurance payout as `230893.33`, drops the payee-unit number, and produces a heavily garbled fee table. Kimi-K2.5 gets the numeric fields right but its fee table is missing most of the 14 items present in the original.

Figure 13: **FinixPhoto Sample 2.** Per-field comparison on a Chongqing outpatient medical receipt. Four fields are highlighted: the invoice number (`0559817`), the Chinese-capital total (肆佰叁拾叁圆捌角整), the payee (龙邱渝), and a synthetic "back-page area" box at the bottom strip of the receipt. FinixDoc gets the invoice number, the Chinese total, and the payee correct, and crucially does not hallucinate any content beyond what is visible on the front of the receipt. Youtu-Parsing renders the Chinese total as a plain "`433.80`" and drops the payee. Both Youtu-Parsing and Kimi-K2.5 hallucinate extra back-page items (e.g., CT, 重症监护, 中医治疗) that are not visible in the input image.

Figure 14: **FinixPhoto Sample 3.** Per-field comparison on a Hubei outpatient medical invoice. Four fields are highlighted: the receipt code (`1420601`), the unified social-credit code (`00000507`), the issue date (`2020-08-12 09:52:59`), and the cashier code (`1890266`). FinixDoc gets all four correct. Youtu-Parsing truncates the receipt code to `20601`, prefixes the social-credit code with an extraneous `123`, drops the issue date, and reads the cashier code as `1890260`. Kimi-K2.5 reads `0000507` (one leading zero missing) for the social-credit code and `2020 06 12 09 52 59` (wrong month, lost separators) for the issue date.