Dolphin: Document Image Parsing via Heterogeneous Anchor Prompting

Anonymous ACL submission

Abstract

Document image parsing is challenging due to its complexly intertwined elements such as text paragraphs, figures, formulas, and tables. Current approaches either assemble specialized expert models or directly generate page-level content autoregressively, facing integration overhead, efficiency bottlenecks, and layout structure degradation despite their decent performance. To address these issues, we present Dolphin (**Do**cument Image **P**arsing via **H**eterogeneous Anchor Prompt**in**g), a novel multimodal document image parsing model following an analyze-then-parse paradigm. In the first stage, Dolphin generates a sequence of layout elements in reading order. These heterogeneous elements, serving as anchors and coupled with task-specific prompts, are fed back to Dolphin for parallel content parsing in the second stage. To train Dolphin, we construct a large-scale dataset of over 30 million samples, covering multi-granularity parsing tasks. Through comprehensive evaluations on both prevalent benchmarks and self-constructed ones, Dolphin achieves state-of-the-art performance across diverse page-level and element-level settings, while ensuring superior efficiency through its lightweight architecture and parallel parsing mechanism. The code and pre-trained models will be made publicly available.

1 Introduction

Document image parsing (Blecher et al.) aims to extract structured contents from images with intertwined elements like text paragraphs, figures, tables, and formulas. As a foundation capability for downstream content analysis (Wang et al., 2024c), it bridges the gap between visual content and machine-readable formats. With the exponential growth of digital documents across domains like academic papers, business reports, and technical documentation, robust document parsing capabilities have become increasingly critical.

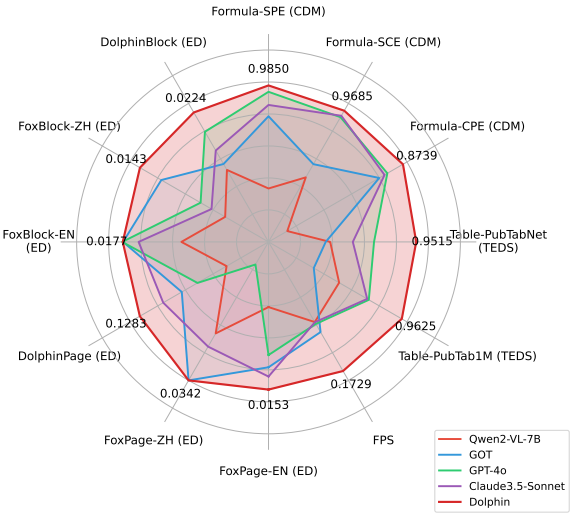


Figure 1: Comparison of Dolphin with advanced VLMs across benchmarks: page-level parsing (plain and complex documents), element-level parsing (text paragraph, table, and formula), and running efficiency (FPS). The outer area represents better performance. Dolphin exhibits the best performance in most evaluations.

Current document image parsing solutions have evolved along two distinct trajectories. The first one (Wang et al., 2024b) integrates specialized models for different OCR tasks (e.g., layout detection, reading order prediction, and recognition for textlines, formulas, tables). These solutions demonstrate strong performance through dedicated expertise, while requiring independent optimization of each model and facing coordination challenges across models. To address these challenges, recent works leverage general or expert vision-language models (VLMs) (Liu et al., 2024b) to directly generate page-level content autoregressively, benefiting from end-to-end training and effective multimodal feature fusion. These methods (Blecher et al.; Kim et al., 2022; Wei et al., 2024b) show impressive results in capturing page-level semantics. However, they also encounter layout structure degradation and efficiency bottlenecks when handling long documents with complex layout.

To synergize the advantages of both approaches while addressing their limitations, we present Dolphin (**Do**cument Image **P**arsing via **H**eterogeneous Anchor Prompt**in**g), a novel vision-language model following an *analyze-then-parse* paradigm. Rather than relying on multiple expert models or purely autoregressive generation, Dolphin decomposes document parsing into two strategic stages. In the first stage, Dolphin performs comprehensive page-level layout analysis by generating element sequence in natural reading order, while preserving rich structural relationships (*e.g.*, figure-caption pairs, table-caption associations, and section title-paragraph hierarchies). These analyzed elements then serve as anchors for the second stage, where element-specific prompts enable efficient parallel parsing of multiple elements. The focused context within each element allows the vision-language model to effectively recognize the document contents.

To train Dolphin on different granularities of tasks, we construct a large-scale dataset of 30 million samples containing both page-level documents and element-level blocks. Note that Dolphin's element-decoupled parsing strategy offers unique advantages in data collection, as acquiring isolated element images (e.g., tables, formulas) and their annotations is more feasible than collecting full document pages with diverse elements.

Comprehensive evaluations are conducted on prevalent benchmarks and self-constructed ones. The results show that Dolphin achieves state-of-the-art performance across diverse page-level and element-level parsing tasks (Figure 1). Moreover, benefiting from its lightweight architecture and element parallel parsing mechanism, Dolphin exhibits considerable advantages in running efficiency.

2 Related Work

Document image parsing enables robust content extraction from rendered document images without relying on source file formats or parsing libraries (e.g., PyMuPDF). Existing solutions can be categorized into two streams: integration-based methods that assemble multiple expert models in a pipeline, and end-to-end approaches that leverage vision-language models to directly generate structured results via autoregressive decoding.

2.1 Integration-based Document Parsing

Traditional document parsing solutions rely on integrating multiple specialized models in a multi-stage pipeline (Xu et al., 2020; Herzig et al., 2020; Zhang et al., 2017). These approaches typically start with layout detection to identify different types of elements (e.g., tables, formulas), followed by dedicated recognizers for each element type. Recent commercial and academic solutions such as Mathpix[1], TextIn[2], and MinerU (Wang et al., 2024b) follow this integration-based paradigm. Notably, MinerU advances this direction by introducing sophisticated content filtering and segmentation strategies. These methods demonstrate strong performance through specialized expertise and have shown great potential in high-precision content extraction. However, they face challenges in system complexity, cross-model coordination, and limited understanding of complex document layouts compared to end-to-end approaches.

2.2 Autoregressive Document Parsing

Recent advances in vision-language models have enabled a new paradigm of end-to-end document image parsing, categorized into two streams.

General VLMs. With the rapid development of large vision-language models, recent works explore applying general-purpose VLMs (Liu et al., 2024b) to document parsing tasks. Models like GPT-4V (Yang et al., 2023), Gemini 1.5 (Team et al., 2024), Qwen2-VL (Wang et al., 2024d), MiniCPM-V2.6 (Yao et al., 2024), DeepSeek-VL2 (Wu et al., 2024), and Step-1V show promising results in document understanding without task-specific training. These models benefit from their large-scale pre-training on diverse visual data, showing strong zero-shot capabilities in document understanding. However, they often face challenges in processing efficiency, specialized element recognition, and layout structure preservation, especially when handling long documents with complex layouts.

Expert VLMs. These models are specifically designed and trained for document parsing or understanding tasks. Nougat (Blecher et al.) pioneers this direction by introducing a encoder-decoder model that converts documents into markup language. GOT (Wei et al., 2024b) introduces an innovative unified model that processes various elements of documents. Other representative works such as Donut (Kim et al., 2022), Wukong-Reader (Bai et al., 2023), KOSMOS (Lv et al., 2023), Vary (Wei et al., 2024a), Fox (Liu et al., 2024a), Monkey (Li et al., 2024), TextHawk (Yu

[1]https://mathpix.com/pdf-conversion/
[2]https://www.textin.ai/

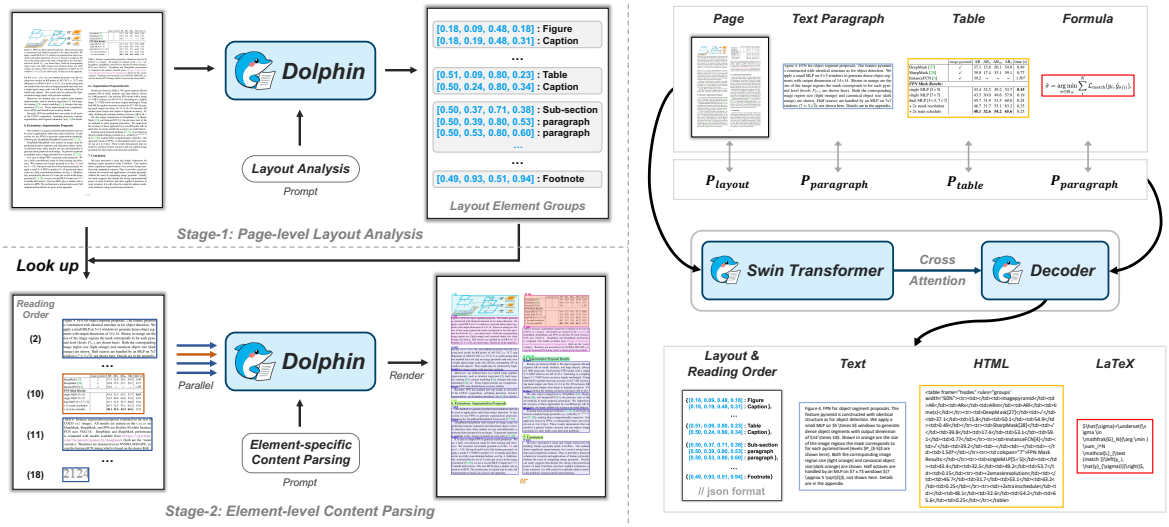


Figure 2: An overview of Dolphin's two-stage document image parsing framework. **Left**: The parsing pipeline consists of Stage 1 for page-level layout analysis that generates structured layout sequences in reading order, and Stage 2 for element-level content parsing. **Right**: Examples of input-output pairs, including page-level layout analysis and element-level content parsing for text paragraphs, tables, and formulas.

et al., 2024), mPLUG-DocOwl (Hu et al., 2024), and PlatPus (Wang et al., 2024e) have been proposed. Despite their impressive performance, they face similar challenges as general VLMs.

3 Approach

In this section, we present our Dolphin in detail. We first provide an overview of our analyze-then-parse architecture, followed by detailed descriptions of the page-level layout analysis stage and element-level content parsing stage.

3.1 Overview

Dolphin follows an analyze-then-parse paradigm built upon an encoder-decoder transformer framework. As shown in Figure 2 (left), given an input document image I, the first stage performs page-level layout analysis to extract elements in reading order. These elements then serve as anchors for the second stage, where type-specific prompts guide parallel parsing of individual elements. The core of both stages is a unified vision-language model, which shares the same parameters but operates on different input granularities with distinct prompting strategies, as presented in Figure 2 (right).

3.2 Page-level Layout Analysis

This stage aims to identify the layout elements and their reading order by the following steps.

Page Image Encoding. We employ Swin Transformer (Liu et al., 2021) as our visual encoder, which takes the page image I as input and outputs a sequence of visual embeddings $z \in \mathbb{R}^{d \times N}$, where d is the embedding dimension and N is the number of image patches. The hierarchical design of Swin enables capturing both global layout patterns and local textual details. Note that the input image is resized and padded to a fixed size of $H \times W$ while preserving its aspect ratio to avoid text distortion.

Layout Sequence Generation. Taking the layout analysis prompt P_{layout} as a guide, the decoder attends to the encoded visual features through the cross-attention mechanism (Vaswani et al., 2017). We adopt mBart (Lewis, 2019) as the decoder. With the prompt "*Parse the reading order of this document.*", the model identifies and arranges document elements sequentially, while preserving structural relationships (e.g., figure-caption pairs, table-caption associations, and section title-paragraph hierarchies). As shown in Figure 2, it generates a sequence of layout elements $L = \{l_1, l_2, ..., l_n\}$, where element l_i specifies its type (e.g., *figure, caption, table, paragraph*) and bounding box. This structured layout sequence provides anchors for the subsequent element-level parsing stage.

3.3 Element-level Content Parsing

The second stage leverages the analyzed layout elements as anchors for parallel element parsing. This design marks a key departure from purely autoregressive approaches, enabling efficient processing while maintaining element-specific expertise. We

Category	Method	Model Size	Plain Doc (ED ↓)		Complex Doc (ED ↓)	Avg. ED	FPS ↑
			Fox-Page-EN	Fox-Page-ZH	Dolphin-Page		
Integration-based	MinerU	1.2B	0.0684	0.07018	0.2813	0.1753	0.0350
	Mathpix	-	**0.0125**	0.0412	0.1635	0.0952	0.0944
Expert VLMs	Nougat	320M	0.1046	0.9918	0.7054	0.6268	0.0673
	Vary-toy	7B	0.082*	-	-	-	-
	Fox	1.8B	0.046*	-	-	-	-
	Kosmos-2.5	1.3B	0.0263	0.2932	0.3920	0.2758	0.0841
	GOT	580M	0.035*	0.038*	0.2497	0.1431	0.0604
General VLMs	InternVL-2.5	8B	0.3000	0.4521	0.4348	0.4054	0.0444
	MiniCPM-o 2.6	8B	0.1590	0.3018	0.3626	0.2965	0.0494
	GLM4v-plus	9B	0.0814	0.1561	0.3826	0.2507	0.0427
	Qwen2-VL-7B	7B	0.1232	0.1638	0.3722	0.2578	0.0315
	Gemini-1.5 pro	-	0.0996	0.0529	0.1958	0.1360	0.0376
	Claude3.5-Sonnet	-	0.0316	0.1327	0.1961	0.1391	0.0320
	GPT-4o-202408	-	0.0585	0.3580	0.2941	0.2512	0.0368
	Step-1v-8k	-	0.0248	0.0401	0.2171	0.1248	0.0417
Ours	**Dolphin**	322M	0.0153	**0.0342**	**0.1283**	**0.0765**	**0.1729**

Table 1: Performance comparison of **page-level document parsing**. "Plain Doc" represents documents containing only text content, while "Complex Doc" includes documents with mixed elements (tables, formulas, and figures). Arrow "↑/↓" indicate whether higher/lower values are better. Results marked with "*" are reported by GOT.

achieve this through two steps:

Element Image Encoding. For each layout element l_i identified in the first stage, we crop its corresponding region from the original image to create a local view I_i. These local views are encoded in parallel using the same Swin Transformer, producing element-specific visual features.

Parallel Content Parsing. With the encoded element features, we employ type-specific prompts to guide the parsing of different elements. As shown in Figure 2 (right), tables employ dedicated prompts P_{table} to parse their HTML format, while formulas share the same prompt $P_{paragraph}$ as text paragraphs since they frequently appear both inline and in display mode within paragraph context, despite their LaTeX markup format. Given the visual feature of the local view I_i and its corresponding prompt p_i, the decoder generates the parsed content in parallel. This parallel processing strategy, combined with element-specific prompting, ensures computational efficiency while maintaining accurate content recognition.

4 Dataset

To enable comprehensive training and evaluation, we construct large-scale datasets spanning multiple document granularities and parsing tasks.

4.1 Training

For training, we collect over 30 million samples covering both page-level documents and element-level components. A comprehensive breakdown of our training dataset, including data sources, granularities, and task, is shown in Table 2. In the following, we describe the preparation and collection of data for different training objectives.

Source	Granularity	#Samples	Task Types
Mixed Documents	Page	0.12M	Layout
HTML	Page	4.37M	Parsing
LaTeX	Page	0.5M	Parsing
Markdown	Page	0.71M	Parsing
Table	Element	1.57M	Parsing
Formula	Element	23M	Parsing
Total	-	30.27M	-

Table 2: Overview of our training data. Note that page-level documents are also decomposed into individual elements for element-specific training.

Mixed Documents. We collect 0.12M documents from diverse sources, including HR materials from HRDoc (Ma et al., 2023), educational materials (exam papers and textbooks), publications (magazines and newspapers), and business documents (presentations and industry reports). All documents are annotated with element-level boundaries and their reading order, enabling training for both layout analysis and order prediction.

HTML. For documents from the HTML source, we utilize dumps from Chinese and English Wikipedia articles to generate synthetic training data through web rendering (Kim et al., 2023). We process HTML content by adding span tags for character-level annotation, and apply random font selection to enhance visual diversity. Through this pipeline, we generate 4.37M page-level samples with comprehensive bounding box annotations at character, word, line and paragraph levels.

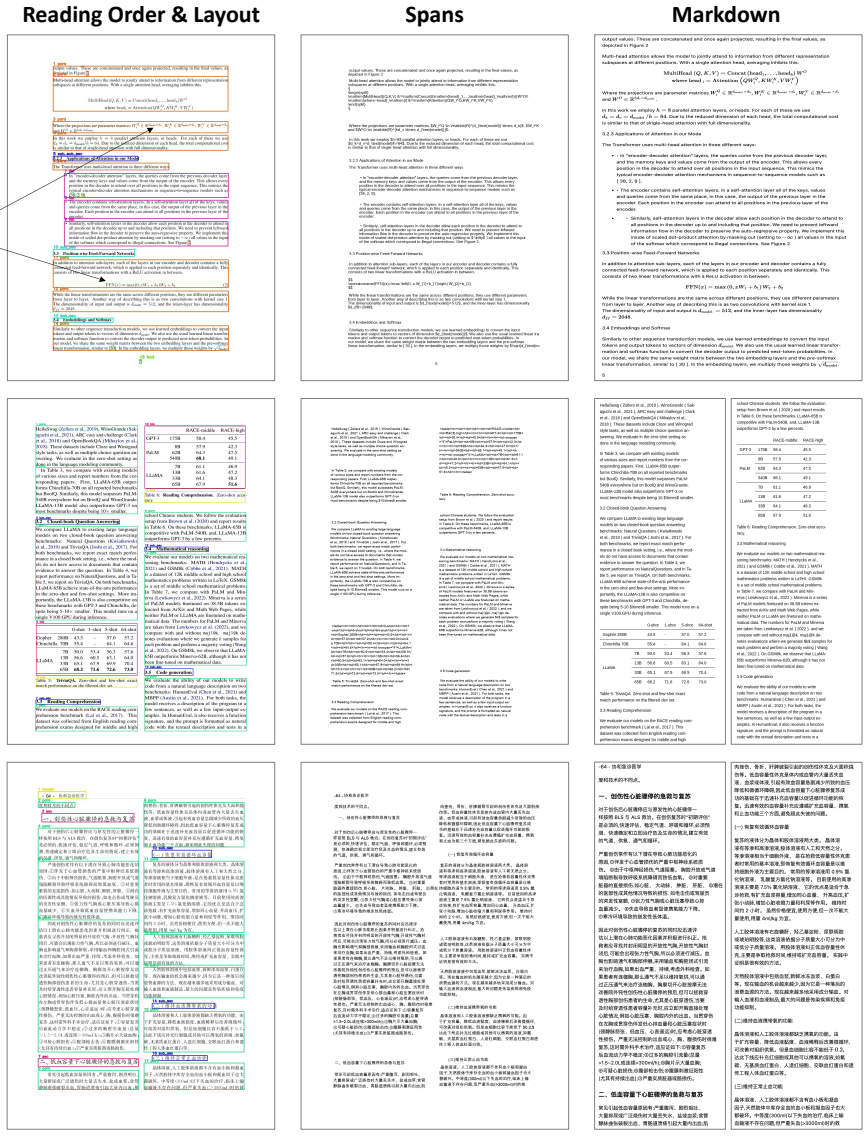


Figure 3: Visualization of Dolphin's **page-level** parsing results. **Left**: Layout analysis form Stage 1 with predicted element boundaries and reading order. **Middle**: Element-specific parsing outputs from Stage 2. **Right**: Final rendered document in markdown format. More cases are shown in the supplementary material.

LaTeX. We collect 0.5M documents from the arXiv database and process them using LaTeX Rainbow (Duan and Bartsch), a specialized rendering framework that preserves document hierarchical structure. This tool renders different element (e.g., formulas, figures) with distinct colors while maintaining the reading order. The rendered documents are then automatically parsed to extract element types, hierarchical relationships, and spatial locations at block, line, and word levels.

Markdown. We collect 0.71M markdown documents from GitHub pages and process them using Pandoc (MacFarlane, 2013) for PDF rendering with several customized templates. Through PyMuPDF-based parsing and content alignment with source markdown, we obtain hierarchical text annotations at paragraph, line, and word levels, as well as some specific element types like tables. Furthermore, we render the formula in different colors and find all formula blocks based on pixel matching.

Tables. For table parsing, we utilize PubTabNet (Zhong et al., 2020) and PubTab1M (Smock et al., 2022), two large-scale datasets of tables extracted from scientific publications. PubTabNet contains 568K tables with HTML annotations, while PubTab1M provides 1M tables with more fine-grained structure annotations.

Formulas. We collect 23M formula expressions in LaTeX format from arXiv sources, including in-line formulas, single-line formulas, and multi-line formulas. The expressions are then rendered formula images using the XeTeX tool. Various

Category	Method	Text Paragraph (ED ↓)			Formula (CDM ↑)			Table (TEDS ↑)	
		Fox-Block		Dolphin-Block	SPE	SCE	CPE	PubTabNet	PubTab1M
		EN	ZH						
Expert Models	UnimerNet-base	-	-	-	**0.9914***	0.94*	0.9595*	-	-
	Mathpix	-	-	-	0.9729*	0.9318*	**0.9671***	-	-
	Pix2tex	-	-	-	0.9619*	0.2453*	0.6489*	-	-
Expert VLMs	TabPedia	-	-	-	-	-	-	**0.9541**	0.9511
	GOT	0.0181	0.0452	0.0931	0.8501	0.7369	0.7197	0.3684	0.3269
General VLMs	GLM-4v-plus	0.0170	0.0400	0.1786	0.9651	0.9585	0.7055	0.5462	0.6018
	Qwen2-VL-7B	0.0910	0.1374	0.1012	0.5339	0.6797	0.1220	0.3973	0.5101
	Gemini-1.5 pro	0.0108	0.0461	0.0857	0.9572	0.9469	0.7171	0.7571	0.7776
	Claude3.5-Sonnet	0.0375	0.1177	0.0746	0.8995	0.9464	0.7543	0.5431	0.7127
	GPT-4o-202408	0.0170	0.1019	0.0489	0.9570	0.9402	0.7722	0.6692	0.7243
	Step-1v-8k	**0.0098**	0.0175	0.0252	0.9526	0.9336	0.7519	0.6808	0.6588
Ours	**Dolphin**	0.0177	**0.0143**	**0.0224**	0.9850	**0.9685**	0.8739	0.9515	**0.9625**

Table 3: Performance comparison of **element-level parsing** across text paragraphs, formulas, and tables. Arrows "↑/↓" indicate whether higher/lower values are better. Results marked with "*" are reported by UnimerNet.

backgrounds and fonts are used in the rendering process to enhance the richness of the images.

4.2 Evaluation

The evaluation is conducted at both page and element levels. At the page level, we evaluate on two distinct benchmarks: Fox-Page (Liu et al., 2024a), which consists of pure text documents, and our constructed Dolphin-Page containing complex documents with interleaved figures, tables, and mathematical formulas. At the element level, we evaluate fine-grained parsing capabilities for text-paragraph, formulas, and tables through the public test sets.

Page-level Evaluation:

(a) Fox-Page. Fox-Page is a bilingual benchmark containing 212 document pages (112 in English and 100 in Chinese) including both single-column and multi-column formats. Each page contains over 1,000 words, making it a challenging testbed for document image parsing.

(b) Dolphin-Page. Our Dolphin-Page is a bilingual benchmark of 211 document pages designed for complex document parsing. It consists of 112 pure text documents and 99 challenging samples with interleaved tables, mathematical formulas, and figures in both single-column and multi-column layouts. All documents are manually annotated with precise transcriptions following the natural reading order, making it a rigorous testbed for evaluating document parsing capabilities.

Element-level Evaluation:

(a) Text Paragraph. For pure text recognition evaluation, we utilize two test sets. The first set follows the official block-level evaluation protocol of Fox-Page (Liu et al., 2024a), containing 424 text paragraph images. The second set is constructed by extracting 1,856 text paragraphs from our Dolphin-Page. Unlike page-level evaluation which considers both reading order prediction and content recognition, this element-level evaluation focuses solely on fundamental text recognition capability.

(b) Formula. For formula recognition evaluation, we utilize three public benchmarks (Wang et al., 2024a) with different complexity levels: SPE with 6,762 simple printed expressions, SCE containing 4,742 screen capture formulas, and CPE consisting of 5,921 complex mathematical expressions. We adopt Character Difference Metric (CDM), which measures the character-level edit distance between predictions and ground truth.

(c) Table. The table recognition evaluation is conducted on two widely-used benchmarks: PubTabNet (Zhong et al., 2020) and PubTab1M (Smock et al., 2022). The test set of PubTabNet contains 7,904 table images from scientific papers, while PubTab1M's test set consists of 10,000 more challenging samples. Both benchmarks evaluate the model's capability in understanding table structures and recognizing cell contents using TEDS (Tree-Edit-Distance-based Similarity) as the metric, which computes the similarity between the predicted and ground-truth HTML table structure.

5 Experiment

5.1 Implementation Details

In the proposed Dolphin, the encoder uses a Swin Transformer with a window size of 7 and hierarchical structure ([2, 2, 14, 2] encoder layers with [4, 8, 16, 32] attention heads). The decoder contains 10 Transformer layers with a hidden dimension of 1024. We train the model using AdamW optimizer

Figure 4: Demonstration of Dolphin's **element-level** parsing across diverse scenarios. Input images are shown in the top row, with corresponding recognition results in the bottom row. **Left**: Text paragraph parsing in complex layouts. **Middle**: Bilingual text paragraph recognition. **Right**: Complex table parsing (rendered results shown).

with a learning rate of 5e-5 and cosine decay schedule. The training is conducted on 32 A100 GPUs for 2 epochs, using a batch size of 16 per device through gradient accumulation.

5.2 Comparison with Existing Methods

Comprehensive evaluations are conducted on both full-page document parsing (plain and complex documents) and individual element recognition tasks (text paragraphs, tables, and formulas).

Page-level Parsing. We evaluate Dolphin's performance on Fox-Page (English and Chinese) and Dolphin-Page benchmarks. As shown in Table 1, despite its lightweight architecture (322M parameters), Dolphin achieves superior performance compared to both integration-based methods and larger VLMs. For pure text documents, Dolphin achieves an edit distances of 0.0153 and 0.0342 on English and Chinese test sets respectively, outperforming specialized VLMs like GOT (with edit distances of 0.035 and 0.038) and general VLMs like GPT-4o (with edit distances of 0.0585 and 0.3580). The advantage becomes more evident on Dolphin-Page, where Dolphin achieves an edit distance of 0.1283, outperforming all baselines in handling documents with mixed elements like tables and formulas. Furthermore, with parallel parsing design, Dolphin demonstrates considerable efficiency gains, achieving 0.1729 FPS, which is nearly 2× faster than the most efficient baseline (Mathpix at 0.0944 FPS).

We visualize three representative cases in Figure 3, showing the complete pipeline from layout analysis (Stage 1) to element-specific parsing (Stage 2), and finally to the rendered document. As demonstrated, Dolphin accurately captures both layout structure and textual content. As shown in Figure 5 (left), Dolphin also exhibits strong text extraction capabilities by accurately parsing content from specified bounding box regions.

Element-level Parsing. Beyond page-level parsing, we conduct extensive experiments to evaluate Dolphin's performance on individual elements, as shown in Table 3. For text paragraph parsing, Dolphin achieves competitive results on both Fox-Block and Dolphin-Block test sets. In formula recognition, Dolphin demonstrates strong capabilities across different complexity levels (SPE, SCE, and CPE), achieving competitive CDM scores comparable to specialized formula recognition methods. For table parsing, our approach shows promising results on both PubTabNet and PubTab1M benchmarks, effectively capturing both structural relationships and cell contents. These consistent strong results across text paragraphs, formulas, and tables demonstrate Dolphin's competitive performance in fundamental recognition tasks.

We further show Dolphin's robustness in Figure 4 through three scenarios: text paragraphs with complex layouts, bilingual text recognition, and structured tables with intricate formats. As shown in Figure 5 (right), Dolphin also supports text spotting by detecting and parsing text lines.

5.3 Ablation Studies

We conduct extensive experiments to validate the effectiveness of the core components in Dolphin.

Parallel Decoding. To investigate the efficiency gains from our parallel decoding strategy in stage 2, we compare our approach with a sequential decoding baseline. As present in Table 4, parallel decoding achieves a 1.8× speedup (0.1729 vs. 0.0971 FPS) while maintaining the same recognition accu-

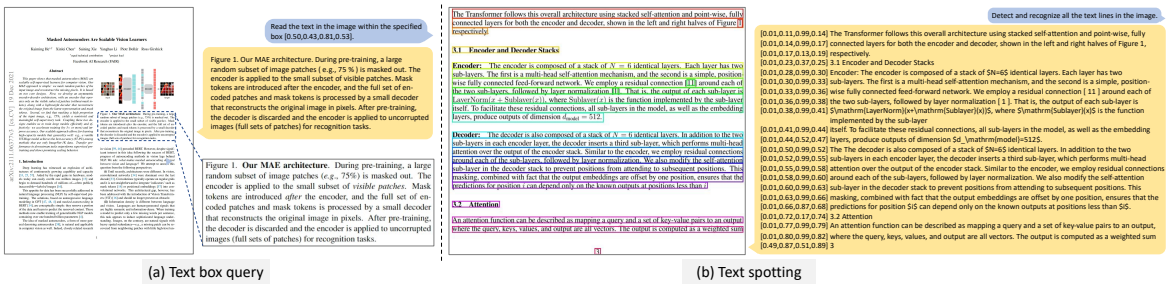


Figure 5: Additional capabilities of Dolphin. **Left**: Parsing the text content from a given bounding box region. **Right**: Text spotting results showing detected text lines (visualized in the image) and their content.

Method	ED ↓	FPS ↑
Dolphin	**0.1283**	**0.1729**
Parallel → Sequential Decoding	-	0.0971
Type-specific → Generic Prompts	0.1613	-
Element Cropping → Box Query	0.1849	-

Table 4: Ablation studies on Dolphin. The first row shows the performance of our full model. The evaluation is conducted on Dolphin-Page dataset.

racy. The speedup is bounded by two factors: (a) the preprocessing overhead for each element before network inference, and (b) the batch size constraint (maximum 16 elements per batch) due to GPU memory limitations, requiring multiple inference passes for documents with numerous elements.

Type-specific vs. Generic Prompts. To investigate the effectiveness of type-specific prompting in the second stage, we compare Dolphin with a baseline variant that uses a generic prompt "*Read the text in the image.*" for all element parsing tasks. As shown in Table 4, our type-specific prompting strategy significantly outperforms the generic baseline (0.1283 vs. 0.1613 in ED). A representative case is shown in Figure 6, where the generic prompt misidentifies a table as a LaTeX formula, while our type-specific prompt successfully parses and renders it. These results demonstrate that incorporating prior knowledge through type-specific prompting effectively improves the model's ability to handle different document elements.

Element Cropping vs. Box Query. To validate our element cropping strategy in the second stage, we compare it with an alternative box query approach that directly prompts the model to recognize elements at specific box (see Figure 5 (left)). As shown in Table 4, our cropping strategy achieves better performance than the box query method. This is likely because cropping provides the model with a focused view of each element, following a "what you see is what you get" principle, while

Input Table Image

encoder	dec. depth	ft acc	hours	speedup
ViT-L, w/ [M]	8	84.2	42.4	-
ViT-L	8	84.9	15.4	2.8×
ViT-L	1	84.8	11.6	**3.7**×
ViT-H, w/ [M]	8	-	119.6†	-
ViT-H	8	85.8	34.5	3.5×
ViT-H	1	85.9	29.3	**4.1**×

Generic Prompt: Misidentified as formula

```
$$\begin{array}{llll}\text { encoder } & \text { dec. depth } & \text { ft acc } & \text { hours } & \text { speedup } \\ \hline \mathrm{ViT}-\mathrm{L}, \mathrm{w} /[\mathrm{M}] & 8 & 84.2 & 42.4 & \text { - } \\ \mathrm{ViT}-\mathrm{L} & 8 & 84.9 & 15.4 & 2.8 \times \\ \mathrm{ViT}-\mathrm{L} & 1 & 84.8 & 11.6 & \mathbf{3. 7} \times \\ \hline \mathrm{ViT}-\mathrm{H}, \mathrm{w} /[\mathrm{M}] & 8 & \text { - } & 119.6^{\dagger} & \text { - } \\ \mathrm{ViT}-\mathrm{H} & 8 & 85.8 & 34.5 & 3.5 \times \\ \mathrm{ViT}-\mathrm{H} & 1 & 85.9 & 29.3 & \mathbf{4. 1} \times\end{array}$$
```

Type-specific Prompt: Correctly parsed as HTML table and successfully rendered

encoder	dec. depth	ft acc	hours	speedup
ViT-L, w/ [M]	8	84.2	42.4	-
ViT-L	8	84.9	15.4	2.8×
ViT-L	1	84.8	11.6	3.7×
ViT-H, w/ [M]	8	-	119.6†	-
ViT-H	8	85.8	34.5	3.5×
ViT-H	1	85.9	29.3	4.1×

Figure 6: A case study demonstrating the effectiveness of type-specific prompts. The generic prompt misidentifies the table as a formula, while our approach correctly parses and renders the table in HTML format.

the box query approach increases task complexity by requiring the model to simultaneously handle location understanding and content recognition.

6 Conclusion

We present Dolphin, a novel document image parsing model that leverages an analyze-then-parse paradigm to address the challenges in document parsing. Our approach first performs page-level layout analysis to generate structured layout elements in reading order, then enables parallel element parsing through heterogeneous anchor prompting. This two-stage design effectively balances efficiency and accuracy, while maintaining a lightweight architecture. Through extensive experiments, we demonstrate Dolphin's strong performance in both page-level and element-level parsing tasks, particularly excelling in handling complex documents with interleaved tables, formulas, and rich formatting in both Chinese and English.

Limitations

Despite Dolphin's promising performance, there are several limitations worth noting. First, our model primarily supports documents with standard horizontal text layout, showing limited capability in parsing vertical text arrangements like ancient manuscripts and rotated tables. Second, while Dolphin handles both Chinese and English documents effectively, its multilingual capacity needs to be expanded. Third, although we achieve efficiency gains through parallel element parsing, there is potential for further optimization through parallel processing of text lines and table cells.

References

Haoli Bai, Zhiguang Liu, Xiaojun Meng, Shuang Liu, LUO Yifeng, Rongfu Zheng, Liangwei Wang, Lu Hou, Jiansheng Wei, Xin Jiang, et al. 2023. Wukong-Reader: Multi-modal pre-training for fine-grained visual document understanding. In *Proceedings of the Annual Meeting Of The Association For Computational Linguistics*.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents. In *Proceedings of the International Conference on Learning Representations*.

Changxu Duan and Sabine Bartsch. LaTex rainbow: Open source document layout semantic annotation framework. In *Proceedings of the Workshop for Natural Language Processing Open Source Software*.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333.

Anwen Hu, Haiyang Xu, Liang Zhang, Jiabo Ye, Ming Yan, Ji Zhang, Qin Jin, Fei Huang, and Jingren Zhou. 2024. mPLUG-DocOwl2: High-resolution compressing for OCR-free multi-page document understanding. *arXiv preprint arXiv:2409.03420*.

Donghyun Kim, Teakgyu Hong, Moonbin Yim, Yoonsik Kim, and Geewook Kim. 2023. On web-based visual corpus construction for visual document understanding. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 297–313.

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

Mike Lewis. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Zhang Li, Biao Yang, Qiang Liu, Zhiyin Ma, Shuo Zhang, Jingxu Yang, Yabo Sun, Yuliang Liu, and Xiang Bai. 2024. Monkey: Image resolution and text label are important things for large multi-modal models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 26763–26773.

Chenglong Liu, Haoran Wei, Jinyue Chen, Lingyu Kong, Zheng Ge, Zining Zhu, Liang Zhao, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. 2024a. Focus anywhere for fine-grained multi-page document understanding. *arXiv preprint arXiv:2405.14295*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024b. Visual instruction tuning. In *Proceedings of the Neural Information Processing Systems*, volume 36.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin Transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10012–10022.

Tengchao Lv, Yupan Huang, Jingye Chen, Yuzhong Zhao, Yilin Jia, Lei Cui, Shuming Ma, Yaoyao Chang, Shaohan Huang, Wenhui Wang, et al. 2023. Kosmos-2.5: A multimodal literate model. *arXiv preprint arXiv:2309.11419*.

Jiefeng Ma, Jun Du, Pengfei Hu, Zhenrong Zhang, Jianshu Zhang, Huihui Zhu, and Cong Liu. 2023. HRDoc: Dataset and baseline method toward hierarchical reconstruction of document structures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 1870–1877.

John MacFarlane. 2013. Pandoc: a universal document converter. *URL: http://pandoc. org*, 8.

Brandon Smock, Rohith Pesala, and Robin Abraham. 2022. PubTables-1M: Towards comprehensive table extraction from unstructured documents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4634–4642.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Neural Information Processing Systems*, pages 5998–6008.

Bin Wang, Zhuangcheng Gu, Guang Liang, Chao Xu, Bo Zhang, Botian Shi, and Conghui He. 2024a. UnimerNet: A universal network for real-world mathematical expression recognition. *arXiv preprint arXiv:2404.15254*.

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

Dongsheng Wang, Natraj Raman, Mathieu Sibue, Zhiqiang Ma, Petr Babkin, Simerjot Kaur, Yulong Pei, Armineh Nourbakhsh, and Xiaomo Liu. 2024c. DocLLM: A layout-aware generative language model for multimodal document understanding. In *Proceedings of the Annual Meeting Of The Association For Computational Linguistics*.

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

Peng Wang, Zhaohai Li, Jun Tang, Humen Zhong, Fei Huang, Zhibo Yang, and Cong Yao. 2024e. PlatyPus: A generalized specialist model for reading text in various forms. In *Proceedings of the European Conference on Computer Vision*, pages 165–183.

Haoran Wei, Lingyu Kong, Jinyue Chen, Liang Zhao, Zheng Ge, Jinrong Yang, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. 2024a. Vary: Scaling up the vision vocabulary for large vision-language model. In *Proceedings of the European Conference on Computer Vision*, pages 408–424.

Haoran Wei, Chenglong Liu, Jinyue Chen, Jia Wang, Lingyu Kong, Yanming Xu, Zheng Ge, Liang Zhao, Jianjian Sun, Yuang Peng, et al. 2024b. General OCR theory: Towards OCR-2.0 via a unified end-to-end model. *arXiv preprint arXiv:2409.01704*.

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, et al. 2024. Deepseek-VL2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. LayoutLM: Pre-training of text and layout for document image understanding. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1192–1200.

Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. 2023. The dawn of LMMs: Preliminary explorations with GPT-4V (ision). *arXiv preprint arXiv:2309.17421*.

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. 2024. MiniCPM-V: A GPT-4V level MLLM on your phone. *arXiv preprint arXiv:2408.01800*.

Ya-Qi Yu, Minghui Liao, Jihao Wu, Yongxin Liao, Xiaoyu Zheng, and Wei Zeng. 2024. TextHawk: Exploring efficient fine-grained perception of multimodal large language models. *arXiv preprint arXiv:2404.09204*.

Jianshu Zhang, Jun Du, Shiliang Zhang, Dan Liu, Yulong Hu, Jinshui Hu, Si Wei, and Lirong Dai. 2017. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*, 71:196–206.

Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. 2020. Image-based table recognition: data, model, and evaluation. In *Proceedings of the European Conference on Computer Vision*, pages 564–580.

In this supplementary material, we provide additional experimental results and implementation details to complement our main paper. Specifically, we present more qualitative results demonstrating Dolphin's parsing capabilities, elaborate on the supported element types, detail our training process, and showcase our synthetic data.

A Qualitative Results

To further demonstrate the superior capabilities of Dolphin, we present comprehensive page-level and element-level parsing results.

Page-level. First, the examples in Figure 8 cover diverse document scenarios, including textbook pages with dense formulas, triple-column English academic papers, and double-column Chinese papers with tables. The results demonstrate that Dolphin can effectively handle documents with different languages, layouts, and element types, maintaining high parsing quality.

Furthermore, we showcase Dolphin's versatility in other text-rich scenarios through Figure 9, where we test the model on mobile phone screenshots, shopping receipts, and webpage captures. These results indicate that Dolphin can accurately capture both the structural layout and textual content in these everyday scenarios.

Element-level. For fine-grained parsing capabilities, we first demonstrate Dolphin's formula recognition in Figure 10, where we evaluate three types of formulas: inline formulas, single-line block formulas, and multi-line block formulas. The results show that Dolphin can accurately parse formulas of varying complexity and layout formats.

We further evaluate Dolphin's table parsing ability in Figure 11, where we test the model on a challenging case containing hundreds of cells. As shown, Dolphin successfully handles this large-scale structured table with precise content recognition and layout preservation.

B Element Design

In this section, we elaborate on Dolphin's supported element types and element-specific parsing strategies through heterogeneous prompting.

Element Types. Our Dolphin supports 16 different types of elements commonly found in document images. Table 5 provides a comprehensive overview of these elements, covering various components from headers to specialized content blocks.

No.	Element	Description
1	title	Paper/document title
2	author	Author names
3	sec	First-level section headings
4	sub_sec	Second-level section headings
5	para	Paragraphs
6	header	Page headers
7	foot	Page footers
8	fnote	Footnotes
9	watermark	Non-content watermarks
10	fig	Figures and images
11	tab	Tables
12	cap	Figure/table captions
13	anno	Figure/table annotations
14	alg	Code blocks/pseudocode
15	list	List-type content
16	reference	References and citations

Table 5: An overview of element types supported by Dolphin. These elements cover the majority of content structures found in documents.

Note that in Stage 1 (page-level layout analysis), we intentionally avoid treating formulas as independent elements. This design choice allows Stage 2 (element-level parsing) to leverage broader contextual information when recognizing mathematical expressions, as formulas are often semantically connected with their surrounding text.

Heterogeneous Anchor Prompting. We summarize the prompts used in Dolphin in Table 6. The first three prompts (page-level layout analysis, text paragraph parsing, and table parsing) are designed for full-page document image parsing, while the latter two (text spotting and text box query) enable additional capabilities for flexible text recognition tasks. Additionally, our Dolphin can also serve as a formula recognition expert model using the text paragraph parsing prompt.

In Stage 2, tables are processed with a dedicated table-specific prompt for structured HTML parsing, while all other elements are treated as text paragraphs and parsed using a unified prompt. This dichotomous design distinguishes structured HTML content from plain text, while also providing robustness against potential element misclassification, as parsing accuracy remains high regardless of element type classification errors.

C Training Details

In this section, we provide more details about Dolphin's training process, including multi-task training strategy, model initialization, and other implementation considerations.

Instruction Tuning. During training phase, we

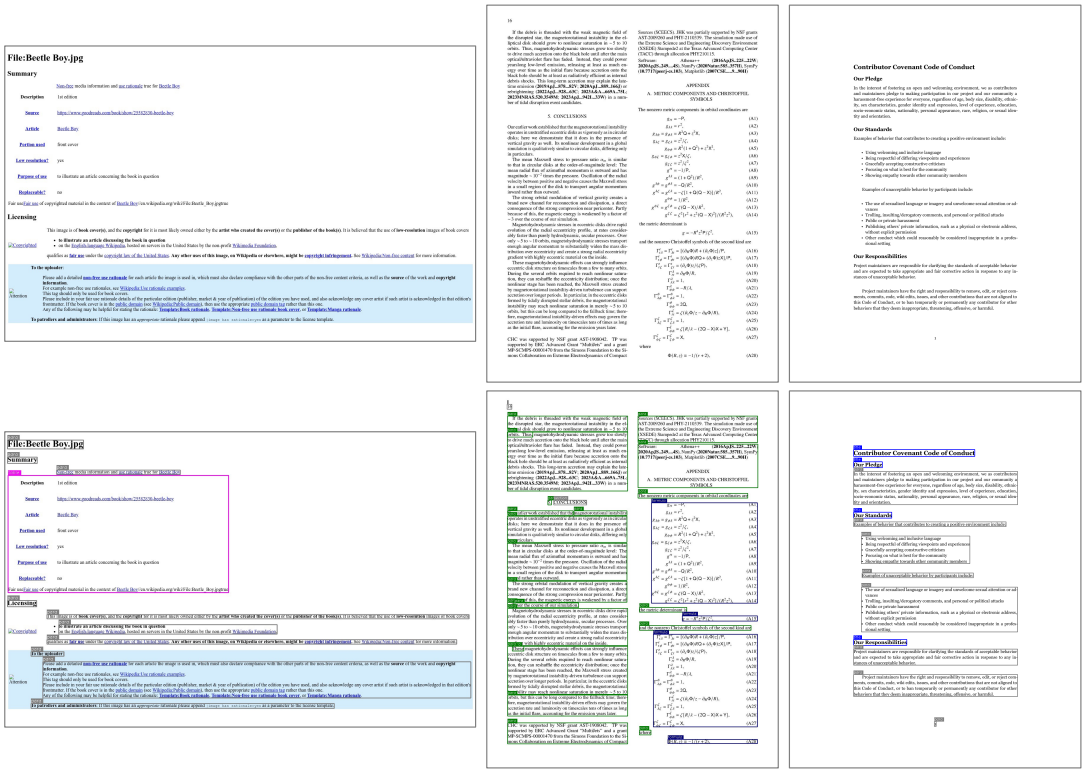


Figure 7: Examples of synthetic training data generated from different source formats. **Top**: rendered document images from HTML (left), LaTeX (middle), and Markdown (right) sources. **Bottom**: corresponding paragraph-level annotations visualized with colored regions.

Task	Prompt
Page-level Layout Analysis	Parse the reading order of this document.
Text Paragraph/Formula Parsing	Read the text in the image.
Table Parsing	Parse the table in the image.
Text Spotting	Detect and recognize all the text lines in the image.
Text Box Query	Read the text in the image within the specified box [x1,y1,x2,y2].

Table 6: Different types of prompts used in Dolphin for document parsing tasks.

adopt a dynamic task selection strategy for our instruction-based framework. Specifically, given a training sample, we randomly select an applicable task from the above five tasks based on its available annotations. This selection is used to construct question-answer pairs. For instance, given a page image with only paragraph-level bounding boxes and content annotations, the available tasks for this sample would include element-level text paragraph parsing and page-level box query parsing.

Model Initialization. We initialize Dolphin with the pretrained weights from Nougat (Blecher et al.), which lacks instruction-following abilities. Then, through our instruction tuning, we extend the model's capabilities to understand and execute diverse prompts, enabling analysis of document layout, reading order, and various textual elements including text paragraphs, tables, and formulas.

Training Loss. Following standard practice in autoregressive language models, we optimize Dolphin using the cross-entropy loss between the predicted token distributions and ground truth ones.

D Synthetic Data Examples

To enrich training data diversity, we synthesize document images from different source formats, including HTML, LaTeX, and Markdown documents. Figure 7 shows three representative examples of our synthetic data. For each format, we show the rendered document (top row) and its corresponding paragraph-level annotations (bottom row).

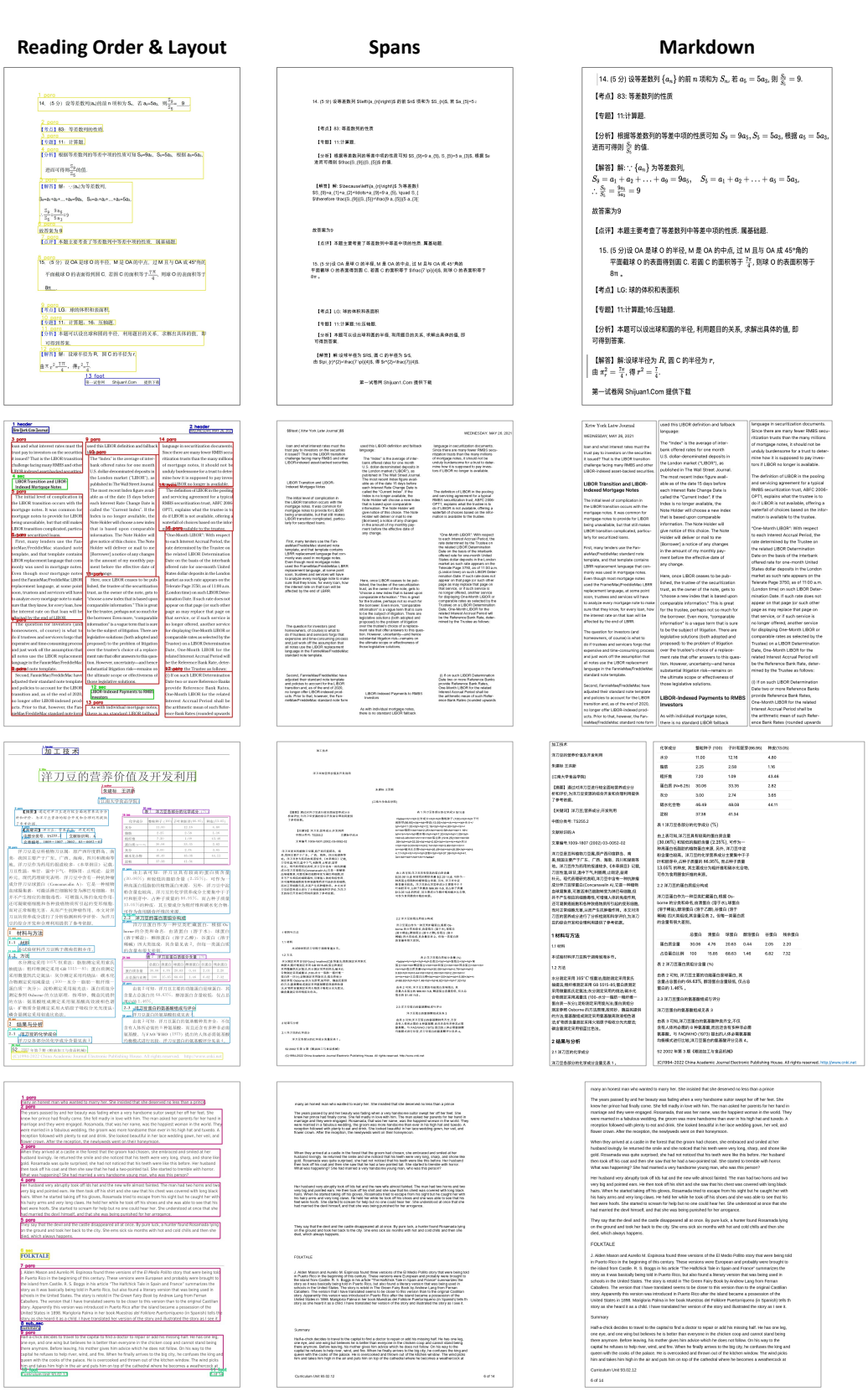
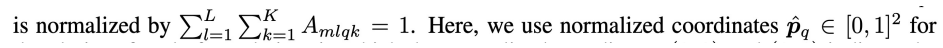


Figure 8: Visualization of Dolphin's **page-level** parsing results. **Left**: Layout analysis form Stage 1 with predicted element boundaries and reading order. **Middle**: Element-specific parsing outputs from Stage 2. **Right**: Final rendered document in markdown format.

Figure 9: Visualization of Dolphin's **page-level** parsing results. **Left**: Input text-rich images including mobile phone screenshots, shopping receipts, and webpage captures. **Middle**: Layout analysis form Stage 1 with predicted element boundaries and reading order. **Right**: Final rendered document in markdown format for the first row, and element-specific parsing outputs from Stage 2 for the second and third rows.

Inline formula image



is normalized by $\sum_{l=1}^{L} \sum_{k=1}^{K} A_{mlqk} = 1$. Here, we use normalized coordinates $\hat{\boldsymbol{p}}_q \in [0,1]^2$ for

Parsing results

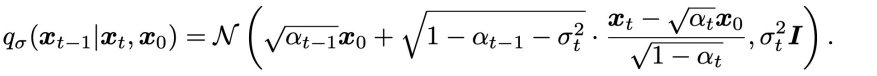
```
is normalized by $\sum_{l=1}^{L} \sum_{k=1}^{K} A_{m l q k}=1$. Here, we use
normalized coordinates $\hat{\boldsymbol{p}}_{q} \in[0,1]^{2}$ for
```

Rendered image

is normalized by $\sum_{l=1}^{L} \sum_{k=1}^{K} A_{mlqk} = 1$. Here, we use normalized coordinates $\hat{\boldsymbol{p}}_q \in [0,1]^2$ for

---

Block formula image



$$q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}}, \sigma_t^2\boldsymbol{I}\right).$$

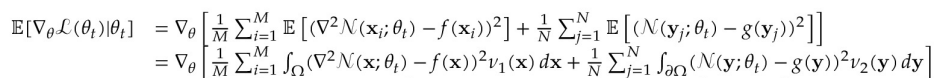
Parsing results

```
$$
q_{\sigma}\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_{t},
\boldsymbol{x}_{0}\right)=\mathcal{N}\left(\sqrt{\alpha_{t-1}}
\boldsymbol{x}_{0}+\sqrt{1-\alpha_{t-1}-\sigma_{t}^{2}} \cdot \frac{\boldsymbol{x}_{t}-
\sqrt{\alpha_{t}} \boldsymbol{x}_{0}}{\sqrt{1-\alpha_{t}}}, \sigma_{t}^{2}
\boldsymbol{I}\right).
$$
```

Rendered image

$$q_\sigma\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}}, \sigma_t^2\boldsymbol{I}\right).$$

Block formula image



$$\begin{aligned}\mathbb{E}[\nabla_\theta \mathcal{L}(\theta_t)|\theta_t] &= \nabla_\theta\left[\tfrac{1}{M}\textstyle\sum_{i=1}^{M}\mathbb{E}\left[(\nabla^2\mathcal{N}(\mathbf{x}_i;\theta_t) - f(\mathbf{x}_i))^2\right] + \tfrac{1}{N}\sum_{j=1}^{N}\mathbb{E}\left[(\mathcal{N}(\mathbf{y}_j;\theta_t) - g(\mathbf{y}_j))^2\right]\right] \\ &= \nabla_\theta\left[\tfrac{1}{M}\textstyle\sum_{i=1}^{M}\int_\Omega(\nabla^2\mathcal{N}(\mathbf{x};\theta_t) - f(\mathbf{x}))^2\nu_1(\mathbf{x})\,d\mathbf{x} + \tfrac{1}{N}\sum_{j=1}^{N}\int_{\partial\Omega}(\mathcal{N}(\mathbf{y};\theta_t) - g(\mathbf{y}))^2\nu_2(\mathbf{y})\,d\mathbf{y}\right] \\ &= \nabla_\theta\left[\int_\Omega(\nabla^2\mathcal{N}(\mathbf{x};\theta_t) - f(\mathbf{x}))^2\nu_1(\mathbf{x})\,d\mathbf{x} + \int_{\partial\Omega}(\mathcal{N}(\mathbf{y};\theta_t) - g(\mathbf{y}))^2\nu_2(\mathbf{y})\,d\mathbf{y}\right] \\ &= \nabla_\theta\mathcal{J}(\mathcal{N}(\cdot;\theta_t))\end{aligned}$$

Parsing results

```
\begin{array} { r l } { \mathbb { E } [ \nabla _ { \theta } \mathcal { L } ( \theta _ { t } ) |
\theta _ { t } ] } & { = \nabla _ { \theta } \left[ \frac { 1 } { M } \sum _ { i = 1 } ^ { M }
\mathbb { E } \left[ ( \nabla ^ { 2 } \mathcal { N } ( \mathbf { x } _ { i } ; \theta _ { t } ) - f
( \mathbf { x } _ { i } ) ) ^ { 2 } \right] + \frac { 1 } { N } \sum _ { j = 1 } ^ { N } \mathbb
{ E } \left[ ( \mathcal { N } ( \mathbf { y } _ { j } ; \theta _ { t } ) - g ( \mathbf { y } _ { j } ) )
^ { 2 } \right] \right] } \\ & { = \nabla _ { \theta } \left[ \frac { 1 } { M } \sum _ { i = 1 } ^
{ M } \int _ { \Omega } ( \nabla ^ { 2 } \mathcal { N } ( \mathbf { x } ; \theta _ { t } ) - f
( \mathbf { x } ) ) ^ { 2 } \nu _ { 1 } ( \mathbf { x } ) \, d \mathbf { x } + \frac { 1 } { N }
\sum _ { j = 1 } ^ { N } \int _ { \partial \Omega } ( \mathcal { N } ( \mathbf { y } ; \theta _
{ t } ) - g ( \mathbf { y } ) ) ^ { 2 } \nu _ { 2 } ( \mathbf { y } ) \, d \mathbf { y } \right] } \\ &
{ = \nabla _ { \theta } \left[ \int _ { \Omega } ( \nabla ^ { 2 } \mathcal { N } ( \mathbf { x } ;
\theta _ { t } ) - f ( \mathbf { x } ) ) ^ { 2 } \nu _ { 1 } ( \mathbf { x } ) \, d \mathbf { x } + \int
_ { \partial \Omega } ( \mathcal { N } ( \mathbf { y } ; \theta _ { t } ) - g ( \mathbf { y } ) ) ^
{ 2 } \nu _ { 2 } ( \mathbf { y } ) \, d \mathbf { y } \right] } \\ & { = \nabla _ { \theta }
\mathcal { J } ( \mathcal { N } ( \cdot ; \theta _ { t } ) ) } \end{array}
```

Rendered image

$$\begin{aligned}\mathbb{E}[\nabla_\theta \mathcal{L}(\theta_t)|\theta_t] &= \nabla_\theta\left[\tfrac{1}{M}\textstyle\sum_{i=1}^{M}\mathbb{E}\left[(\nabla^2\mathcal{N}(\mathbf{x}_i;\theta_t) - f(\mathbf{x}_i))^2\right] + \tfrac{1}{N}\sum_{j=1}^{N}\mathbb{E}\left[(\mathcal{N}(\mathbf{y}_j;\theta_t) - g(\mathbf{y}_j))^2\right]\right] \\ &= \nabla_\theta\left[\tfrac{1}{M}\textstyle\sum_{i=1}^{M}\int_\Omega(\nabla^2\mathcal{N}(\mathbf{x};\theta_t) - f(\mathbf{x}))^2\nu_1(\mathbf{x})\,d\mathbf{x} + \tfrac{1}{N}\sum_{j=1}^{N}\int_{\partial\Omega}(\mathcal{N}(\mathbf{y};\theta_t) - g(\mathbf{y}))^2\nu_2(\mathbf{y})\,d\mathbf{y}\right] \\ &= \nabla_\theta\left[\int_\Omega(\nabla^2\mathcal{N}(\mathbf{x};\theta_t) - f(\mathbf{x}))^2\nu_1(\mathbf{x})\,d\mathbf{x} + \int_{\partial\Omega}(\mathcal{N}(\mathbf{y};\theta_t) - g(\mathbf{y}))^2\nu_2(\mathbf{y})\,d\mathbf{y}\right] \\ &= \nabla_\theta\mathcal{J}(\mathcal{N}(\cdot;\theta_t))\end{aligned}$$

Figure 10: Visualization of Dolphin's **formula** parsing results. From top to bottom, we show three formula types: **inline formula**, **single-line block formula**, and **multi-line block formula**. For each case, we visualize the complete parsing pipeline: input formula image (top), LaTeX parsing output (middle), and rendered formula (bottom). These results demonstrate Dolphin's capability to accurately parse formulas of varying complexity.

Shots	Method	AR	BG	DE	EL	EN	ES	FR	HI	RU	SW	TH	TR	UR	VI	ZH	Avg.
1	FT	33.2	33.3	33.3	33.1	33.3	33.2	32.8	33.0	33.3	33.0	33.3	32.9	32.9	33.2	33.2	33.1
	SP	35.4	36.4	36.4	36.6	36.5	37.6	37.9	36.0	37.5	34.1	35.9	34.7	35.0	35.5	**36.7**	36.1
	PCT	33.2	35.4	34.8	35.1	35.9	35.3	35.7	34.6	36.2	33.8	34.6	34.3	33.1	34.9	35.0	34.8
	MPT	**37.0**	**38.5**	**37.8**	**38.1**	**38.6**	**38.1**	**38.7**	**37.2**	**38.5**	**36.5**	**37.1**	**37.6**	**37.3**	**37.9**	35.7	**37.6**
2	FT	33.5	33.3	33.7	33.3	34.1	33.5	33.7	33.2	33.5	33.3	33.8	33.6	33.5	34.0	33.3	33.5
	SP	36.6	37.9	38.0	38.2	38.0	38.0	38.3	36.2	38.9	34.3	37.5	34.6	35.2	37.2	36.7	37.0
	PCT	34.1	39.0	39.1	38.2	39.9	40.6	40.5	37.9	39.9	36.5	37.2	36.9	34.7	37.9	37.1	38.0
	MPT	**41.6**	**42.8**	**40.8**	**43.2**	**43.2**	**42.5**	**42.8**	**40.4**	**43.3**	**36.8**	**40.5**	**41.0**	**41.1**	**41.4**	**38.2**	**41.3**
4	FT	34.2	34.5	34.1	34.3	34.1	34.1	34.5	34.0	34.3	33.7	34.0	34.0	34.1	34.2	34.2	34.1
	SP	37.4	39.7	39.2	39.7	40.2	38.9	40.5	37.1	40.6	35.3	38.1	35.3	36.9	37.2	38.9	38.3
	PCT	33.9	37.2	37.0	36.2	37.0	37.7	37.5	36.4	37.4	34.2	34.7	34.7	33.5	35.0	35.6	35.9
	MPT	**42.9**	**43.6**	**44.3**	**43.6**	**45.5**	**44.2**	**44.1**	**42.8**	**44.1**	**40.2**	**43.4**	**42.7**	**42.4**	**43.8**	**43.1**	**43.4**
8	FT	32.8	32.7	32.8	32.9	32.7	32.6	33.0	33.3	32.7	33.0	33.2	33.0	33.1	32.5	32.4	32.8
	SP	37.4	39.6	38.1	39.1	40.0	38.8	39.2	36.5	40.3	35.6	38.5	35.3	36.5	37.8	37.1	38.0
	PCT	40.2	40.6	40.9	41.7	41.9	41.7	41.6	41.0	40.6	39.2	41.4	**41.4**	38.4	41.3	**41.2**	40.9
	MPT	**42.7**	**43.0**	**41.9**	**42.4**	**43.1**	**42.3**	**42.1**	**40.8**	**42.6**	**39.4**	**41.9**	40.1	**40.7**	**42.2**	40.2	**41.7**
16	FT	33.6	33.4	33.3	33.5	34.1	33.4	33.3	33.4	33.4	33.6	33.6	33.4	33.5	33.3	33.5	33.5
	SP	39.5	39.9	39.1	40.4	41.1	40.2	40.4	37.4	40.7	37.1	39.3	36.5	36.0	38.2	38.3	38.9
	PCT	**43.6**	40.8	36.9	**45.7**	**46.5**	41.5	44.3	**44.8**	42.4	40.1	**43.9**	**43.7**	**42.5**	**44.7**	**44.8**	43.1
	MPT	43.5	**43.8**	**44.0**	43.9	45.2	**44.2**	**44.3**	42.9	**43.4**	**40.2**	42.5	41.8	42.0	43.4	42.2	**43.1**
32	FT	36.1	36.3	35.7	35.7	36.5	36.2	36.0	35.5	35.9	35.0	35.6	36.0	35.4	36.1	36.3	35.9
	SP	41.7	43.4	42.8	42.3	44.9	42.9	43.3	39.2	43.5	37.7	40.2	41.1	39.8	43.0	39.8	41.7
	PCT	45.7	45.4	44.4	**47.4**	49.6	45.5	**48.8**	**46.7**	45.5	40.3	41.6	44.3	42.9	46.7	45.6	45.4
	MPT	**47.1**	**47.6**	**47.9**	47.1	**49.9**	**49.1**	48.2	46.3	**47.3**	**43.3**	**47.2**	**47.2**	**45.3**	**49.0**	**47.1**	**47.3**
64	FT	41.4	41.2	41.5	40.7	42.6	41.4	40.8	41.2	40.2	40.6	40.7	41.4	40.5	41.7	41.0	41.1
	SP	43.9	44.2	47.5	45.1	50.5	47.9	48.6	41.8	43.7	41.3	45.9	45.3	42.6	47.6	45.1	45.4
	PCT	48.1	50.2	49.3	50.6	51.1	50.9	51.3	47.6	49.1	44.6	47.3	47.4	44.0	49.7	48.2	48.6
	MPT	**50.7**	**52.7**	**53.1**	**52.2**	**55.4**	**53.8**	**53.1**	**50.2**	**51.0**	**46.2**	**51.5**	**50.4**	**49.1**	**53.0**	**52.3**	**51.7**
128	FT	43.9	44.4	44.4	43.7	46.3	44.6	44.5	42.9	42.7	41.7	43.0	43.2	42.7	44.9	43.8	43.8
	SP	46.2	46.8	47.8	47.6	53.0	48.5	49.6	47.3	45.5	41.7	47.5	46.4	44.5	45.6	48.7	47.1
	PCT	50.4	51.9	52.8	53.4	55.0	53.8	53.3	51.5	51.7	47.0	50.0	50.9	47.9	51.7	51.2	51.5
	MPT	**53.2**	**56.1**	**56.0**	**55.4**	**57.4**	**56.4**	**56.6**	**53.5**	**54.8**	**48.6**	**54.0**	**53.1**	**51.8**	**55.2**	**55.4**	**54.5**
256	FT	53.3	55.6	56.5	55.0	58.8	56.9	56.4	52.5	53.6	50.5	52.6	53.8	51.3	55.0	53.0	54.3
	SP	52.7	55.2	49.6	53.7	59.5	55.0	55.3	50.6	51.4	46.5	53.4	46.1	44.9	52.8	51.5	51.9
	PCT	54.7	56.7	56.3	57.9	60.3	58.3	58.3	54.6	55.2	**51.6**	55.6	54.6	52.6	57.4	55.8	56.0
	MPT	**59.0**	**61.1**	**60.9**	**60.6**	**65.8**	**63.0**	**61.9**	**57.6**	**60.6**	50.7	**59.2**	**57.8**	**56.1**	**60.7**	**60.8**	**59.7**

Shots	Method	AR	BG	DE	EL	EN	ES	FR	HI	RU	SW	TH	TR	UR	VI	ZH	Avg.
1	FT	33.2	33.3	33.3	33.1	33.3	33.2	32.8	33.0	33.3	33.0	33.3	32.9	32.9	33.2	33.2	33.1
	SP	35.4	36.4	36.4	36.6	36.5	37.6	37.9	36.0	37.5	34.1	35.9	34.7	35.0	35.5	36.7	36.1
	PCT	33.2	35.4	34.8	35.1	35.9	35.3	35.7	34.6	36.2	33.8	34.6	34.3	33.1	34.9	35.0	34.8
	MPT	37.0	38.5	37.8	38.1	38.6	38.1	38.7	37.2	38.5	36.5	37.1	37.6	37.3	37.9	35.7	37.6
2	FT	33.5	33.3	33.7	33.3	34.1	33.5	33.7	33.2	33.5	33.3	33.8	33.6	33.5	34.0	33.3	33.5
	SP	36.6	37.9	38.0	38.2	38.0	38.0	38.3	36.2	38.9	34.3	37.5	34.6	35.2	37.2	36.7	37.0
	PCT	34.1	39.0	39.1	38.2	39.9	40.6	40.5	37.9	39.9	36.5	37.2	36.9	34.7	37.9	37.1	38.0
	MPT	41.6	42.8	40.8	43.2	43.2	42.5	42.8	40.4	43.3	36.8	40.5	41.0	41.1	41.4	38.2	41.3
4	FT	34.2	34.5	34.1	34.3	34.1	34.1	34.5	34.0	34.3	33.7	34.0	34.0	34.1	34.2	34.2	34.1
	SP	37.4	39.7	39.2	39.7	40.2	38.9	40.5	37.1	40.6	35.3	38.1	35.3	36.9	37.2	38.9	38.3
	PCT	33.9	37.2	37.0	36.2	37.0	37.7	37.5	36.4	37.4	34.2	34.7	34.7	33.5	35.0	35.6	35.9
	MPT	42.9	43.6	44.3	43.6	45.5	44.2	44.1	42.8	44.1	40.2	43.4	42.7	42.4	43.8	43.1	43.4
8	FT	32.8	32.7	32.8	32.9	32.7	32.6	33.0	33.3	32.7	33.0	33.2	33.0	33.1	32.5	32.4	32.8
	SP	37.4	39.6	38.1	39.1	40.0	38.8	39.2	36.5	40.3	35.6	38.5	35.3	36.5	37.8	37.1	38.0
	PCT	40.2	40.6	40.9	41.7	41.9	41.7	41.6	41.0	40.6	39.2	41.4	41.4	38.4	41.3	41.2	40.9
	MPT	42.7	43.0	41.9	42.4	43.1	42.3	42.1	40.8	42.6	39.4	41.9	40.1	40.7	42.2	40.2	41.7
16	FT	33.6	33.4	33.3	33.5	34.1	33.4	33.3	33.4	33.4	33.6	33.6	33.4	33.5	33.3	33.5	33.5
	SP	39.5	39.9	39.1	40.4	41.1	40.2	40.4	37.4	40.7	37.1	39.3	36.5	36.0	38.2	38.3	38.9
	PCT	43.6	40.8	36.9	45.7	46.5	41.5	44.3	44.8	42.4	40.1	43.9	43.7	42.5	44.7	44.8	43.1
	MPT	43.5	43.8	44.0	43.9	45.2	44.2	44.3	42.9	43.4	40.2	42.5	41.8	42.0	43.4	42.2	43.1
32	FT	36.1	36.3	35.7	35.7	36.5	36.2	36.0	35.5	35.9	35.0	35.6	36.0	35.4	36.1	36.3	35.9
	SP	41.7	43.4	42.8	42.3	44.9	42.9	43.3	39.2	43.5	37.7	40.2	41.1	39.8	43.0	39.8	41.7
	PCT	45.7	45.4	44.4	47.4	49.6	45.5	48.8	46.7	45.5	40.3	41.6	44.3	42.9	46.7	45.6	45.4
	MPT	47.1	47.6	47.9	47.1	49.9	49.1	48.2	46.3	47.3	43.3	47.2	47.2	45.3	49.0	47.1	47.3
64	FT	41.4	41.2	41.5	40.7	42.6	41.4	40.8	41.2	40.2	40.6	40.7	41.4	40.5	41.7	41.0	41.1
	SP	43.9	44.2	47.5	45.1	50.5	47.9	48.6	41.8	43.7	41.3	45.9	45.3	42.6	47.6	45.1	45.4
	PCT	48.1	50.2	49.3	50.6	51.1	50.9	51.3	47.6	49.1	44.6	47.3	47.4	44.0	49.7	48.2	48.6
	MPT	50.7	52.7	53.1	52.2	55.4	53.8	53.1	50.2	51.0	46.2	51.5	50.4	49.1	53.0	52.3	51.7
128	FT	43.9	44.4	44.4	43.7	46.3	44.6	44.5	42.9	42.7	41.7	43.0	43.2	42.7	44.9	43.8	43.8
	SP	46.2	46.8	47.8	47.6	53.0	48.5	49.6	47.3	45.5	41.7	47.5	46.4	44.5	45.6	48.7	47.1
	PCT	50.4	51.9	52.8	53.4	55.0	53.8	53.3	51.5	51.7	47.0	50.0	50.9	47.9	51.7	51.2	51.5
	MPT	53.2	56.1	56.0	55.4	57.4	56.4	56.6	53.5	54.8	48.6	54.0	53.1	51.8	55.2	55.4	54.5
256	FT	53.3	55.6	56.5	55.0	58.8	56.9	56.4	52.5	53.6	50.5	52.6	53.8	51.3	55.0	53.0	54.3
	SP	52.7	55.2	49.6	53.7	59.5	55.0	55.3	50.6	51.4	46.5	53.4	46.1	44.9	52.8	51.5	51.9
	PCT	54.7	56.7	56.3	57.9	60.3	58.3	58.3	54.6	55.2	51.6	55.6	54.6	52.6	57.4	55.8	56.0
	MPT	59.0	61.1	60.9	60.6	65.8	63.0	61.9	57.6	60.6	50.7	59.2	57.8	56.1	60.7	60.8	59.7

Figure 11: Visualization of Dolphin's **table** parsing capability. **Top**: Input large-scale table image containing hundreds of cells. **Bottom**: Rendered HTML table based on Dolphin's parsing result. This example demonstrates Dolphin's strong ability in handling large-scale structured table images.