Dolphin: Document Image Parsing via Heterogeneous Anchor Prompting

Hao Feng*, Shu Wei*, Xiang Fei*, Wei Shi*†, Yingdong Han, Lei Liao,
Jinghui Lu, Binghong Wu, Qi Liu, Chunhui Lin, Jingqun Tang, Hao Liu, Can Huang†

ByteDance

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 limitations, 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 are publicly available at https://github.com/ByteDance/Dolphin

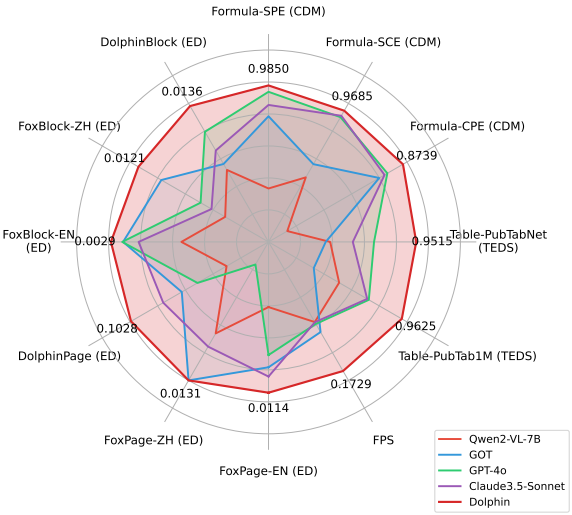

Figure 1: Comparison of Dolphin with advanced VLMs across benchmarks: page-level parsing (plain and complex documents), element-level parsing (text paragraph, table, and formula), and running efficiency (FPS). The outer area represents better performance. Dolphin exhibits the best performance in most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

1 Introduction

Document image parsing (Blecher et al.) aims to extract structured content from images containing intertwined elements such as text paragraphs, figures, tables, and formulas. As a foundational 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.

Current document image parsing solutions have evolved along two distinct trajectories. The first one (Wang et al., 2024b) integrates specialized models for different OCR (Tang et al., 2022b; Zhao et al., 2024b; Tang et al., 2022a) tasks (*e.g.*, layout detection, reading order prediction, and recognition for textlines, formulas, and tables). These solutions demonstrate strong performance through dedicated expertise, but require independent optimization of each model and face coordination challenges across components. 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

*The first four authors contributed equally to this work.
†Corresponding author

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 parsing long documents with complex layouts.

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 an 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. Notably, 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 parallel element 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., 2020b; Herzig et al., 2020; Zhang et al., 2017). These approaches typically begin 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 significant potential in high-precision content extraction. However, they face challenges in system complexity, cross-model coordination, and limited understanding of complex document layouts when 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, researchers have begun exploring the application of general-purpose VLMs (Liu et al., 2024b) to document parsing tasks. Models such as GPT-4V (Yang et al., 2023), Claude-series[3], Gemini-series (Team et al., 2024), QwenVL-series (Wang et al., 2024d; Bai et al., 2025), MiniCPM-series (Yao et al., 2024), InternVL-series (Chen et al., 2024), DeepSeek-VL2 (Wu et al., 2024), and Step-1V demonstrate promising results in document understanding without task-specific training. These models benefit from large-scale pre-training on diverse visual data, exhibiting strong zero-shot capabilities. However, they frequently face challenges in processing efficiency, specialized element recognition, and layout structure preservation, particularly when processing long documents with complex layouts.

Expert VLMs. These models are specifically designed and trained for document parsing or understanding tasks. Nougat (Blecher et al.) pioneered

[1]https://mathpix.com/pdf-conversion/
[2]https://www.textin.ai/
[3]https://www.anthropic.com/news/claude-3-5-sonnet

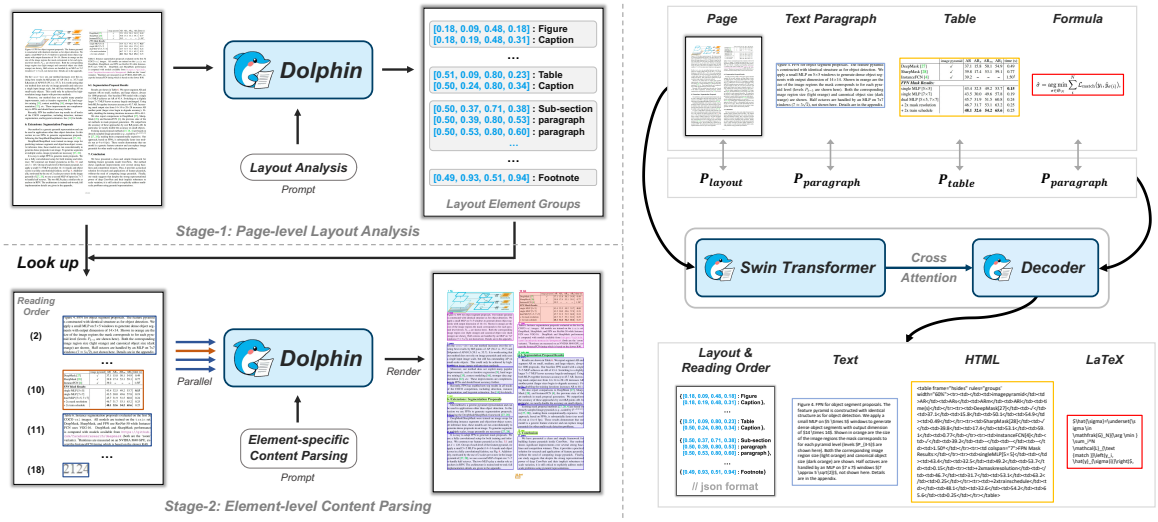

Figure 2: Overview of Dolphin's two-stage document image parsing paradigm. **Left**: The pipeline consists of Stage 1 for page-level layout analysis that generates structured layout sequences in reading order, and Stage 2 for element-level content parsing. **Right**: Examples of input-output pairs, including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

this direction by introducing an encoder-decoder model that converts documents into markup language. GOT (Wei et al., 2024b) presented an innovative unified model that processes various document elements. Other representative work such as Donut (Kim et al., 2022), LayoutLM-series (Xu et al., 2020b,a; Huang et al., 2022), UDOP (Tang et al., 2023), Wukong-Reader (Bai et al., 2023), KOSMOS-series (Lv et al., 2023; Peng et al., 2023), UniDoc (Feng et al., 2023), UReader (Ye et al., 2023b), DocPedia (Feng et al., 2024), TGDoc (Wang et al., 2023), Vary (Wei et al., 2024a), Fox (Liu et al., 2024a), Monkey-series (Li et al., 2024; Liu et al., 2024c), TabPedia (Zhao et al., 2024a), TextSquare (Tang et al., 2024a), DocFusion (Chai et al., 2024), TextHawk-series (Yu et al., 2024a,b), mPLUG-DocOwl-series (Ye et al., 2023a; Hu et al., 2024), SmolDocling (Nassar et al., 2025), PlatPus (Wang et al., 2024e), olmOCR (Poznanski et al., 2025), Ocean-OCR (Chen et al., 2025), and Mistral-OCR[4] have been proposed. Despite their impressive performance, these expert VLMs 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 paradigm, followed by detailed descriptions of the page-level layout analysis stage and element-level content parsing stage.

[4]https://mistral.ai/fr/news/mistral-ocr

3.1 Overview

Dolphin follows an analyze-then-parse paradigm built upon an encoder-decoder transformer architecture. 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 through 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).

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	0.0685	0.0702	0.2770	0.1732	0.0350
	Mathpix	-	0.0126	0.0412	0.1586	0.0924	0.0944
Expert VLMs	Nougat	250M	0.1036	0.9918	0.7037	0.6131	0.0673
	Kosmos-2.5	1.3B	0.0256	0.2932	0.3864	0.2691	0.0841
	Vary	7B	0.092*	0.113*	-	-	-
	Fox	1.8B	0.046*	0.061*	-	-	-
	GOT	580M	0.035*	0.038*	0.2459	0.1411	0.0604
	olmOCR	7B	0.0235	0.0366	0.2000	0.1148	0.0427
	SmolDocling	256M	0.0221	0.7046	0.5632	0.4636	0.0140
	Mistral-OCR	-	0.0138	0.0252	0.1283	0.0737	0.0996
General VLMs	InternVL-2.5	8B	0.3000	0.4546	0.4346	0.4037	0.0444
	InternVL-3	8B	0.1139	0.1472	0.2883	0.2089	0.0431
	MiniCPM-o 2.6	8B	0.1590	0.2983	0.3517	0.2882	0.0494
	GLM4v-plus	9B	0.0814	0.1561	0.3797	0.2481	0.0427
	Gemini-1.5 pro	-	0.0996	0.0529	0.1920	0.1348	0.0376
	Gemini-2.5 pro	-	0.0560	0.0396	0.2382	0.1432	0.0231
	Claude3.5-Sonnet	-	0.0316	0.1327	0.1923	0.1358	0.0320
	GPT-4o-202408	-	0.0585	0.3580	0.2907	0.2453	0.0368
	GPT-41-250414	-	0.0489	0.2549	0.2805	0.2133	0.0337
	Step-1v-8k	-	0.0248	0.0401	0.2134	0.1227	0.0417
	Qwen2-VL	7B	0.1236	0.1615	0.3686	0.2550	0.0315
	Qwen2.5-VL	7B	0.0135	0.0270	0.2025	0.1112	0.0343
Ours	**Dolphin**	322M	**0.0114**	**0.0131**	**0.1028**	**0.0575**	**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. **Boldface** indicates the best performance and underlined values denote the second-best.

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 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

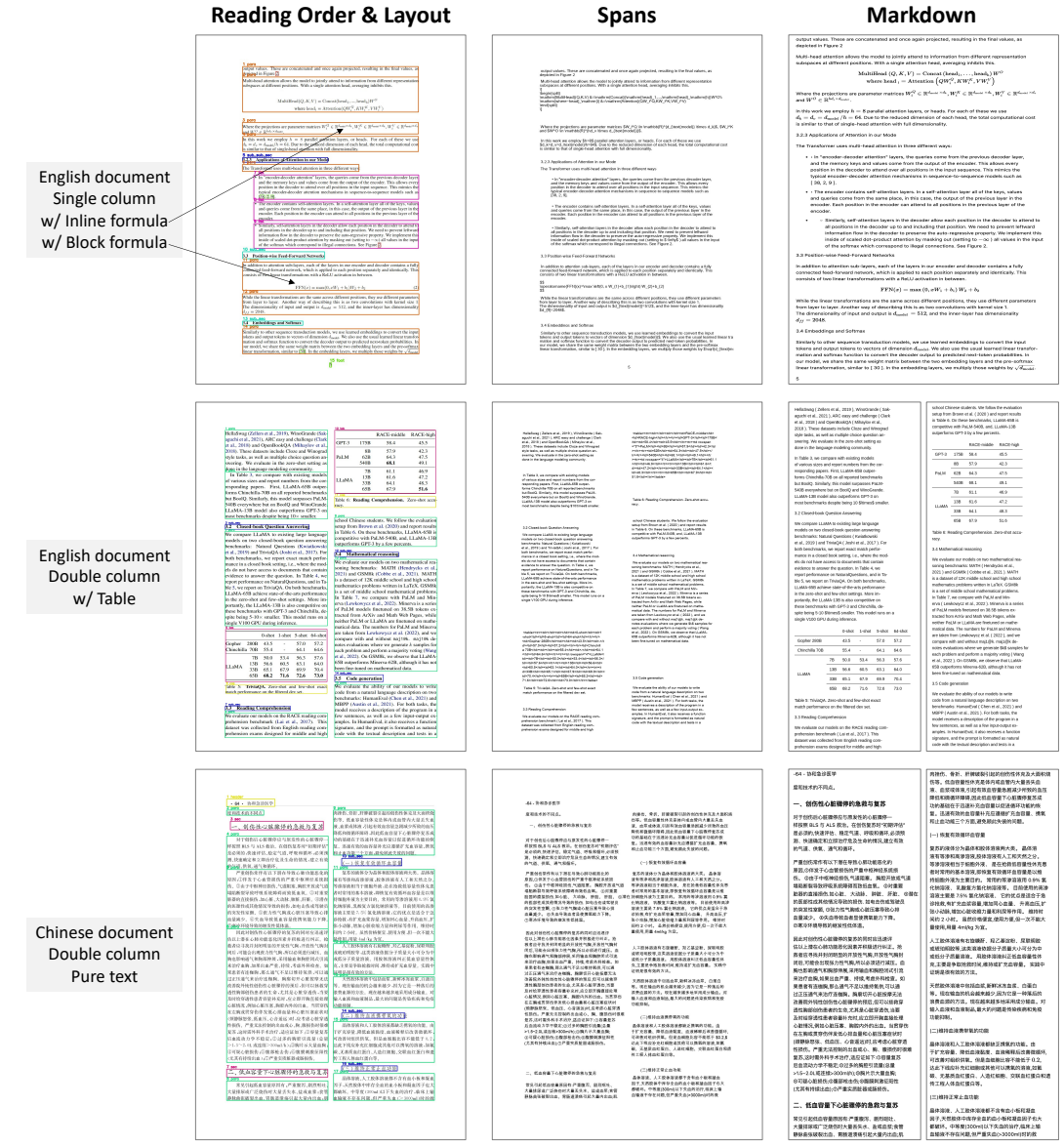

Figure 3: Visualization of Dolphin's **page-level** parsing results. **Left**: Layout analysis form Stage 1 with predicted element boundaries and reading order. **Middle**: Element-specific parsing outputs from Stage 2. **Right**: Final rendered document in markdown format. More cases are shown in the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

of data for different training objectives.

Mixed Documents. We collect 0.12M documents from diverse sources, including 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

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.

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
	Qwen2.5-VL-7B	0.0803	0.0301	0.0712	0.9486	0.9484	0.8309	0.6169	0.6462
	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**	**0.0029**	**0.0121**	**0.0136**	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.

character, word, line and paragraph levels.

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 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 the page and the element levels. At the page level, we evaluate the models on two distinct benchmarks: Fox-Page (Liu et al., 2024a), which consists of pure text documents, and our newly constructed Dolphin-Page containing complex documents with interleaved figures, tables, and mathematical formulas. At the element level, we assess the 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 210 document pages designed for complex document parsing. It consists of 111 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