SmolDocling: An ultra-compact vision-language model for end-to-end multi-modal document conversion

Ahmed Nassar[1], Andres Marafioti[2], Matteo Omenetti[1], Maksym Lysak[1], Nikolaos Livathinos[1], Christoph Auer[1], Lucas Morin[1], Rafael Teixeira de Lima[1], Yusik Kim[1], A. Said Gurbuz[1], Michele Dolfi[1], Miquel Farré[2], Peter W. J. Staar[1]

IBM Research[1], HuggingFace[2]

https://huggingface.co/ds4sd/SmolDocling-256M-preview
Version 1.0, March 14th 2025

Abstract

We introduce SmolDocling, an ultra-compact vision-language model targeting end-to-end document conversion. Our model comprehensively processes entire pages by generating DocTags, a new universal markup format that captures all page elements in their full context with location. Unlike existing approaches that rely on large foundational models, or ensemble solutions that rely on handcrafted pipelines of multiple specialized models, SmolDocling offers an end-to-end conversion for accurately capturing content, structure and spatial location of document elements in a 256M parameters vision-language model. SmolDocling exhibits robust performance in correctly reproducing document features such as code listings, tables, equations, charts, lists, and more across a diverse range of document types including business documents, academic papers, technical reports, patents, and forms — significantly extending beyond the commonly observed focus on scientific papers. Additionally, we contribute novel publicly sourced datasets for charts, tables, equations, and code recognition. Experimental results demonstrate that SmolDocling competes with other Vision Language Models that are up to 27 times larger in size, while reducing computational requirements substantially. The model is currently available, datasets will be publicly available soon.

1 Introduction

For decades, converting complex digital documents into a structured, machine-processable format has been a significant technical challenge. This challenge primarily stems from the substantial variability in document layouts and styles, as well as the inherently opaque nature of the widely used PDF format, which is optimized for printing rather than semantic parsing. Intricate layout styles and visually challenging elements such as forms, tables, and complex charts can significantly impact the reading order and general understanding of documents. These problems have driven extensive research and development across multiple domains of computer science. On one hand, sophisticated ensemble systems emerged, which decompose the conversion problem into several sub-tasks (e.g. OCR, layout analysis, table structure recognition, classification) and tackle each sub-task independently. Although such systems can achieve high-quality results for many document types while maintaining relatively low computational demands, they are often difficult to tune and generalize.

On the other hand, in the recent past, great interest has developed around large foundational multimodal models that can solve the whole conversion task in one shot, while simultaneously offering flexible querying and parametrization through prompts. This approach has been made possible by the advent of multimodal pre-training of large vision-language models (LVLMs), which opened up a vast array of opportunities for leveraging diverse data sources, including PDF documents. However, the literature on this topic highlights a significant gap in the availability of high-quality and open-access datasets suitable for training robust multi-modal models for the task of document understanding. Furthermore, relying on LVLMs may introduce common issues associated with such models, including hallucinations and the use of significant computational resources, making them impractical from both a quality and cost perspective.

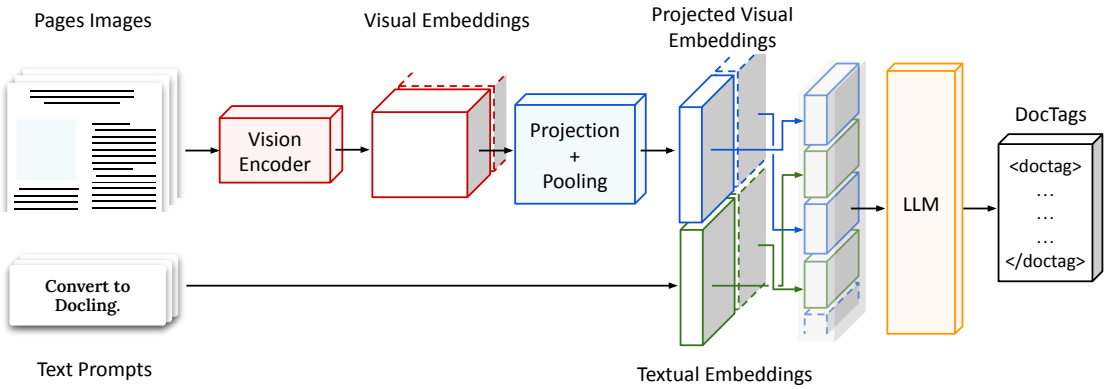


Figure 1: **SmolDocling/SmolVLM architecture.** SmolDocling converts images of document pages to *DocTags* sequences. First, input images are encoded using a vision encoder and reshaped via projection and pooling. Then, the projected embeddings are concatenated with the text embeddings of the user prompt, possibly with interleaving. Finally, the sequence is used by an LLM to autoregressively predict the *DocTags* sequence.

In this work, we outline how we close the gaps left by publicly available datasets and establish a training approach to achieve end-to-end, full-featured document conversion through a vision-language model, which we name *SmolDocling*. Our main contributions can be summarized as follows:

- SmolDocling: An ultra-compact VLM for end-to-end document conversion, based on Hugging Face's most recent SmolVLM-256M [56], which is between 5 and 10 times smaller in parameters than comparable VLMs tuned on document understanding tasks. Hence, we reduce the computational complexity to roughly the same order of magnitude as typical representatives of ensemble methods. Uniquely, SmolDocling learns and produces a unified document representation that captures content, structure and spatial location of all elements in a page.
- We augment existing document pre-training datasets with additional feature annotations, and create new datasets for tasks with insufficient coverage in open multi-modal training datasets, including code listings, equations, and chart understanding. Importantly, we construct *full-page* ground truth data that combines key annotation features such as layout, table structure, code listings, charts, and formulas in their natural context, and derive instruction datasets from all training datasets used.
- We introduce *DocTags*, a document markup format optimized to efficiently represent the full content and layout characteristics of a document, inspired by [55].
- We evaluate the achieved prediction quality of SmolDocling on several tasks, including comparisons to popular community models.

2 Related Work

2.1 Large Vision-Language Models (LVLMs)

With the advent of large language models (LLMs) [74, 66, 82, 32, 90, 4, 8, 25, 2] characterized by their powerful contextual understanding and adaptability, several approaches have emerged to extend them into large vision-language models. Some proprietary, closed-source LLMs have been adapted into multimodal large language models (MLLMs), such as GPT-4o [65], Gemini [80], and Claude 3.5 [5], demonstrating exceptional capabilities across various modalities, including vision. Open-source methods are actively advancing to match the capabilities of proprietary MLLMs. BLIP-2 [47] is one of the earliest works combining a vision encoder with a frozen LLM (OPT [102] or FlanT5 [14]) using a lightweight transformer (Q-former [101]). Building upon BLIP-2, MiniGPT-4 [105] integrated a frozen visual encoder with a frozen LLM (Vicuna [85]) using a Q-Former network and a single projection layer. LLaVA [52, 51] employs a similar architecture but uses a minimal adapter layer. LLaVA-OneVision [45] and LLaVA-NeXT-Interleave [46] build upon LLaVA to support multi-image, higher resolution, and video understanding. For handling high resolutions and text-image compositions and comprehension, InternLM-XComposer introduces a family of models [99, 100, 20, 19] designed specifically for these tasks. Qwen-VL [8] introduces a position-aware adapter to address efficiency issues caused by the vision encoder generating long image feature sequences. Qwen2.5-VL [9]

employs windowed attention with 2D rotary positional embeddings to process native-resolution inputs efficiently and integrates an vision-language merger for dynamic feature compression.

In this work, we build upon the SmolVLM architecture approach [56]. Precisely, we adopt SmolVLM-256M, a compact variant of base SmolVLM-2.2B model, inspired by the architecture of Idefics3 [41] which employs the shape-optimized SigLIP encoder as its visual backbone. It applies an aggressive pixel shuffle strategy to compress visual features, reduces the number of image hidden states and introduces new special tokens to separate sub-images and improve tokenization efficiency.

2.2 State of the Art in Document Understanding

Several powerful document understanding solutions have emerged on the market in the past decade as commercial cloud offerings on hyperscalers [24, 62, 3, 6], frontier models such as GPT-4o [65] or Claude [5], or as open-source libraries such as Docling [53], Grobid [1], Marker [67], MinerU [86] or Unstructured [84]. Typical tasks that document understanding encompasses are document classification [96, 29], OCR [48], layout analysis [95, 70], table recognition [103, 55, 78], key-value extraction [29, 35, 31], chart understanding [49, 57, 50, 26, 60, 93, 94, 13, 58, 33, 34], equation recognition [77, 68, 18, 97, 23], and more.

Ensemble systems, such as the aforementioned open-source libraries [53, 1, 67, 86], implement a pipeline in source code, which conditionally applies specialized, single-task models on a given input document, composing the prediction results into a meaningful document representation. Typically, each task entails human-crafted handling of pre-processing and post-processing logic (i.e. setting confidence thresholds for layout detections, matching layout elements to text cells, conditionally applying models).

Conversely, multi-task models aim at providing a single model capable of handling various document-understanding related tasks simultaneously, benefiting from shared context and shared representations learned across these different tasks. OCR-reliant methods, such as LayoutLM [96, 95, 29] and UDOP [79], use text extracted from an external OCR engine, along with image and text bounding box locations, as input. In contrast, OCR-free methods such as Donut [35], Dessurt [17], DocParser [76], Pix2Struct [43] are transformer-based models trained end-to-end to take an image as input and directly output text. Large Vision Language Models (LVLMs) such as LLaVA [51], LLaVA-OneVision [45], UReader [98], Kosmos-2 [69], and Qwen-VL [8] all leverage various vision encoders, projection adapters, and LLMs. While they may use different image patching techniques and instruction-tuning datasets to handle a range of vision tasks, they commonly evaluate their document understanding capabilities on datasets like DocVQA [59] and mPLUG-DocOwl 1.5 datasets [27], primarily focusing on question answering and reasoning.

Our work focuses primarily on document understanding tasks related to conversion and structural recognition, aiming to represent documents with high accuracy and completeness. The closest works to our approach are Nougat [12], DocOwl 2 [28], GOT [89], and Qwen2.5-VL [9]. DocOwl 2 and GOT are dedicated to precise conversion and recognition of document structures. On one hand, DocOwl 2 employs a dynamic shape-adaptive cropping module to process high-resolution images, feeding them into a ViT vision encoder. A dedicated module compresses the processed visual features before feeding them into a LLaMa-based LLM. On the other hand, GOT focuses on converting diverse elements—such as text, formulas, molecular diagrams, tables, charts, sheet music, and geometric shapes—into structured formats. It achieves a lightweight architecture by utilizing a SAM vision encoder [37], a linear projection layer, and a Qwen-0.5B LLM. Building on these approaches, our work not only extends the range of tasks but also broadens the types of documents handled, moving beyond a strictly scientific focus to support a more diverse array of document formats. In parallel, Qwen2.5-VL introduces the *Omni-Parsing* strategy, that integrates multiple document elements into a unified HTML-based representation. This format encodes layout bounding box information and reading order for various document elements such as paragraphs and charts.

3 SmolDocling

Building on Hugging Face's SmolVLM [56], SmolDocling converts document page images into sequences representing their content and structure. It produces a unified page representation (DocTags markup) that is optimized for the LLM backbone and interpretable by Docling [53]. This new DocTags format covers the reproduction of captions, charts, forms, code, equations, tables, footnotes, lists, page footers and headers, section headings, and text. SmolDocling both recognizes the type and location of these elements as well as performs OCR within them. Our approach enables full-page conversion with reading-order and hierarchical linking between elements (e.g., linking captions with figures and tables), providing bounding boxes for each component, supporting isolated predictions on individual elements (e.g., cropped tables) and grouping for lists and nested lists.

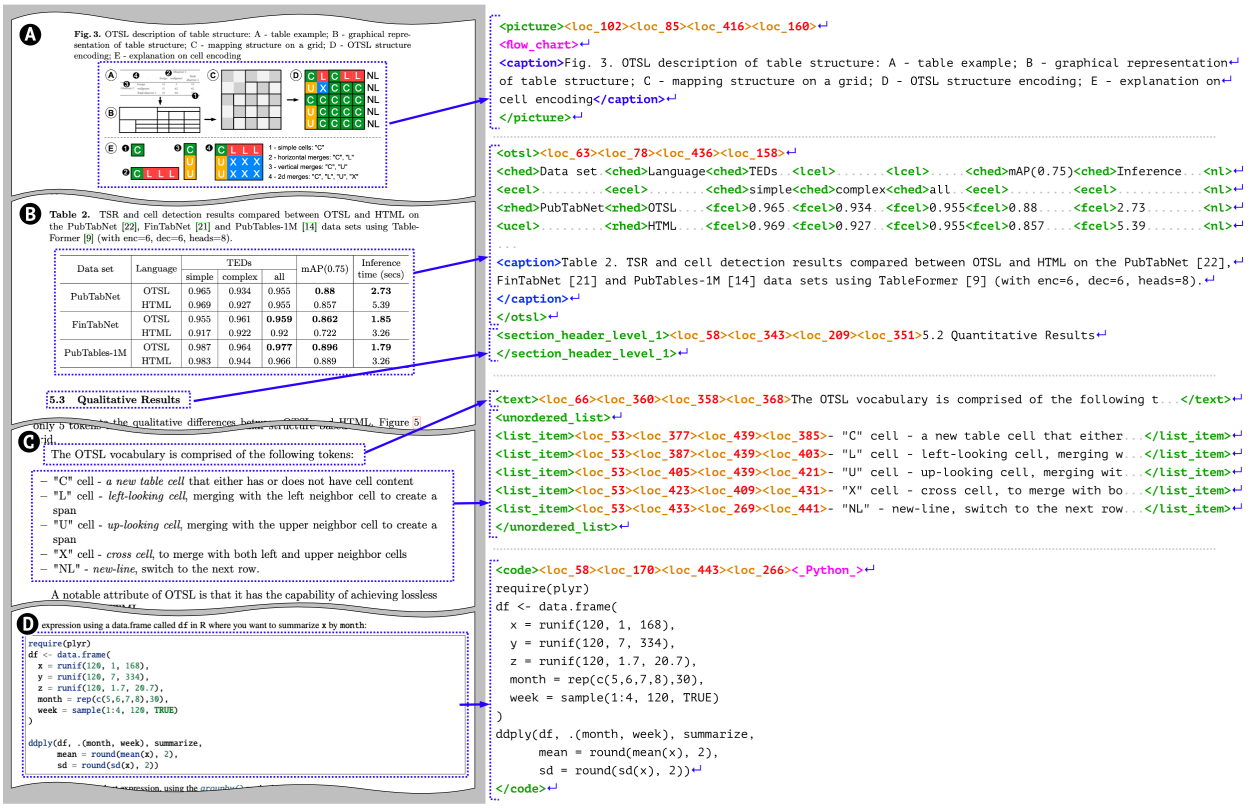


Figure 2: **DocTags format** describes key features of document elements: type of an element (text, picture, table, code, etc.), location on the page, and content. Nested tags convey additional information: pictures and tables may nest captions, table structure is represented in OTSL tags, lists nest list items, code and pictures carry a classification. All *DocTags* output is taken from SmolDocling predictions, artificial line-breaks and dots (...) were inserted for better readability. A, B, C - snippets of various pages from [55], D - from [92]

3.1 Model Architecture

The model architecture is illustrated in Figure 1. SmolVLM-256M relies on SigLIP base patch-16/512 (93M) as visual backbone and compared to the 2.2B version of the same model, its training data was rebalanced to emphasize document understanding (41%) and image captioning (14%), combining The Cauldron [42], Docmatix [41] datasets with the addition of MathWriting [23]. It uses lightweight variant of the SmolLM-2 family (135M) [2] as the language backbone and employs a radical pixel shuffle method that compresses each 512x512 image patch into 64 visual tokens. Last but not least, tokenization efficiency was also improved in SmolVLM-256M by increasing the pixel-to-token ratio to 4096 pixels per token and introducing special tokens for sub-image separators.

3.2 DocTags

To train VLMs for standardized document conversion, we introduce the *DocTags* markup format, illustrated in Figure 3. Inspired by OTSL [55], *DocTags* define a structured vocabulary of unambiguous tags and rules that explicitly separate textual content from document structure. This disentanglement improves performances of Image-to-Sequence models by minimizing confusions. In contrast, direct conversion to standard formats such as HTML or Markdown is often lossy and ambiguous, lacking explicit structural markup, failing to preserve page layout context, and increasing total tokens count. Pieces of text content that represent basic elements in *DocTags* are enclosed in XML-style tags. Tags define document block types such as: *text, caption, footnote, formula, title, list_item, page_footer, page_header, picture, section_header, otsl, code, document_index, etc.* Each element can nest additional location tags encoding its position on the page as a bounding box, represented in DocTags as <loc_*x1*><loc_*y1*><loc_*x2*><loc_*y2*>.

Special blocks, such as tables and images, nest additional descriptors for captions, tabular structure, or image classes. To encode table structures, we fully include the OTSL vocabulary into *DocTags*. OTSL has the advantage of being minimalist and supporting cell spans and header information. Its tokens are interleaved with text, allowing

simultaneous encoding of both table structure and content. In order to promote robust visual-semantic alignment in our document understanding pipeline, we maintain a uniform *DocTags* representation for cropped page elements (e.g., tables, code, equations) that is identical to their full-page counterparts. This consistency ensures that the model can leverage similar visual cues whether an element is presented in isolation or in the context of an entire page, thereby strengthening the learning signal for both computer vision and downstream machine learning tasks.

For a complete list of tags and detailed description of the format, consult the appendix.

3.3 Model Training

We employ a curriculum learning approach to progressively align our model for document conversion, ensuring faster convergence. As an initial step, we incorporate DocTags as tokens in our tokenizer. To align the LLM part we freeze the vision encoder and train only the remaining network to adapt it to the new output format which it hasn't seen before. To ensure comprehensive coverage of all DocTags, we maintain a balanced mix of tasks and data types during training. Next, we unfreeze the vision encoder and train the model on our pretraining datasets (see Sec. 4.1), along with all task-specific conversion datasets, including tables, code, equations, and charts. Finally, we fine-tune using all available datasets.

4 Data

For document understanding tasks, each page image represents the visual layout of textual elements—such as paragraphs, formulas, tables, code, or figures with embedded textual content. Publicly available datasets that unify all these characteristics remain limited, and existing corpora often suffer from small sample sizes or fragmented formats. In this section, we explain how we mitigate these issues in both pre-training and task-specific training scenarios.

4.1 Pre-training datasets

In the domain of document understanding, currently available open-access document pre-training datasets include OCR-IDL [10], IIT-CDIP [44], CCpdf [83], and WordScape [88]. OCR-IDL and IIT-CDIP, industry-sourced from the 1990s, provide only page images and OCR text. CCpdf and WordScape, derived from Common Crawl, focus on raw PDFs and Word documents, respectively. While WordScape enriches documents with layout, table structure, and topic information via XML markup, its visual variability is limited. Notably, none of these datasets offer rich annotations for all page elements.

DocLayNet-PT. Given the lack of public multimodal document pre-training data with sufficient annotation features, we constructed a new document pre-training dataset named DocLayNet-PT. DocLayNet-PT is a 1.4M pages dataset extracted from the DocFM dataset [25] of unique PDF documents, sourced from CommonCrawl, Wikipedia, and business-related documents. We focused on this dataset to look for documents including visually distinct content such as equations, tables, code and charts with colorful layouts. This dataset was augmented with weak annotations through a series of processing steps. First, we applied PDF parsing to all pages to obtain the text layout including positional information using Docling [7, 53]. Next, we performed an enrichment step to obtain an enhanced, machine-friendly representation of each page. Specifically, we provide annotations for layout elements, table structure, language, topic, and figure classification on each page in DocTag format.

Docmatix. To preserve SmolVLM's original DocVQA capabilities, we adapted the same weak annotation strategy used for DocLayNet-PT and applied it to the 1.3M documents inside Docmatix dataset [41]. In addition to the 9.5 million DocVQA questions, we introduced another instruction requiring the full conversion of multi-page documents to DocTags.

4.2 Task-specific Datasets

Layout. To optimize prediction quality for document layout and table structure, we sampled 76K pages from DocLayNet-PT. Each page underwent human annotation and rigorous quality review, resulting in a high-quality ground-truth dataset. Of these, 60K pages were utilized for fine-tuning, constituting what we refer to as DocLayNet v2. Additionally, we extracted 63K pages from WordScape, specifically filtering for pages containing both text and tables, and utilized their inherently structure-preserving format as a reliable source of ground truth. To further enhance the model's adaptability to diverse layouts, colors, and fonts, we synthetically generated 250K pages using content from Wikipedia, creating the dataset known as SynthDocNet. This synthetic dataset enables the controlled generation of various document types with precise annotations. All datasets share a unified annotation format and class definitions, ensuring consistency and comprehensive document representation.

Tables. For the table structure recognition task we trained our model on public datasets such as PubTables-1M [78], FinTabNet [103], WikiTableSet [54] and tabular information extracted from WordScape documents. We transformed table structure information from the original ground-truth to OTSL format [55]. Then, we interleaved each cell tag with ground-truth text. This yields a condensed sequence which represents the structure of the table (including merged cells, and header declaration) along with its textual content.

Charts. For chart reconstruction, several datasets are publicly available [58, 33, 34, 93] that consist of both synthetic and web-scraped charts. However, these datasets are limited by either the quantity of real-world data or the visual diversity of synthetic examples. This has prompted recent studies [50, 26, 60, 94, 93, 13, 57] to crawl or render additional datasets, which unfortunately are rarely shared publicly. To address these gaps, we generated our own chart dataset comprising four types of charts: line, pie, bar, and stacked bar. Using data from 90,000 tables sourced from the FinTabNet dataset [103], we generated a comprehensive dataset containing 5,000 line charts, 380,000 pie charts, 380,000 bar charts, and 77,000 stacked bar charts. Each data point was rendered using three different visualization libraries (details provided in the Appendix), yielding a total of 2.5 million visually diverse charts.

Code. Technical books and scientific documents frequently include code snippets, making it essential for document-processing models to accurately interpret code structure, particularly indentation, which carries semantic significance in certain programming languages. Several publicly available datasets provide extensive collections of code snippets with permissive licenses in text format [38, 71], typically aggregated from sources like GitHub. However, a common limitation is that these datasets contain only text-based code snippets, lacking any image-based representation. To address this gap, we leveraged LaTeX and Pygments [16] to generate visually diverse renderings of code (details provided in the Appendix). Our dataset includes 9.3M million rendered code snippets generated at 120 dpi, covering 56 programming languages.

Equations. Accurately parsing mathematical formulas is crucial for document-processing models, as formulas often contain nuanced structures and semantics critical for scientific understanding. To effectively train our model for formula transcription, we combined publicly available datasets with a custom-rendered dataset. The public datasets [18, 22, 11, 23] collectively comprise approximately 730k unique formulas. In addition, we extracted 4.7 million formulas from arXiv by applying a regular expression to the source LaTeX code. The extracted formulas underwent a rigorous normalization procedure to ensure the model was trained only on correct and standardized LaTex code. Subsequently, these normalized equations were visually rendered using LaTeX (details provided in the Appendix). Overall, our dataset includes 5.5 million unique formulas, each rendered at a resolution of 120 dpi.

Dataset Name	Size	Type
TheCauldron [42]	2.63M	Multi Task
DocLayNet-PT-Instruct	1.62M	
DocMatix w/ Doc Conversion* [41]	952K	
SynthChartNet*	2.5M	Chart
PlotQA [61]	225K	
FigureQA [33]	100K	
Unichart [57]	600K	
ChartQA [58]	28K	
DocLayNet-PT	1.40M	Conversion
Doc4MLlaVa - RVLCDIP	742K	
SynthDocNet	375K	
DocLayNet [70]	81K	
DocLayNet v2	76K	
WordScape-PT	63K	
PubTable1M [78]	1M	Table
WordScape Tables [88]	207K	
WikiTableSet (EN) [54]	204K	
FinTabNet [103]	90K	
SynthFormulaNet*	5.5M	Formula
SynthCodeNet*	9.3M	Code

Table 1: **Datasets.** Overview of the datasets used for the training and evaluation of SmolDocling. Datasets we contribute as part of this publication are denoted with (*).

4.3 Document Instruction Tuning datasets

To reinforce the recognition of different page elements and introduce document-related features and no-code pipelines, we leveraged rule-based techniques and the Granite-3.1-2b-instruct [25] LLM. Using samples from DocLayNet-PT pages, we generated one instruction by randomly sampling layout elements from a page. These instructions included tasks such as "Perform OCR at bbox," "Identify page element type at bbox," and "Extract all section headers from the page." In addition, we train with Cauldron [42] to avoid catastrophic forgetting with the amount of conversation datasets introduced.

Method	Size	Edit Distance ↓	F1-score ↑	Precision ↑	Recall ↑	BLEU ↑	METEOR ↑
Full-page							
Qwen2.5 VL [9]	7B	0.56	0.72	0.80	0.70	0.46	0.57
GOT [89]	580M	0.61	0.69	0.71	0.73	0.48	0.59
Nougat (base) [12]	350M	0.62	0.66	0.72	0.67	0.44	0.54
SmolDocling (Ours)	256M	**0.48**	**0.80**	**0.89**	**0.79**	**0.58**	**0.67**
Code listings							
SmolDocling (Ours)	256M	0.11	0.92	0.94	0.91	0.87	0.89
Equations							
Qwen2.5 VL [9]	7B	0.22	0.89	0.91	0.87	0.68	0.77
GOT [89]	580M	**0.11**	**0.95**	0.95	**0.96**	**0.85**	**0.91**
Nougat (base) [12]	350M	0.62	0.60	0.60	0.53	0.33	0.41
SmolDocling (Ours)	256M	**0.11**	**0.95**	**0.96**	0.95	0.83	0.89

Table 2: **Structured document text recognition.** We evaluate OCR performance and document formatting on DocLayNet, focusing on textual elements, excluding tables. Text accuracy is measured through multiple metrics (Edit Distance, F1-Score, Precision, Recall, BLEU and METEOR) as proposed in [12].

5 Experiments

In this section, we assess the performance of SmolDocling on document understanding tasks, and compare the results to previously published models and baselines. Capturing the qualities and defects of SmolDocling and other models in comparable numbers is a non-trivial effort for reasons we outline further below. Therefore, we report results on standard evaluation tasks and metrics independently (see section 5.2), but also provide a qualitative assessment of output characteristics (see section 5.3).

5.1 Implementation Details

Our training was conducted using 64 NVIDIA A100 80GB GPUs, with each epoch requiring 38 hours. The model was trained for a total of 4 epochs using the AdamW optimizer with a learning rate of 2×10^{-4} and 2×10^{-6} for the vision encoder once unfrozen. We also employed gradient clipping to 1.0 for stability and a warmup ratio of 0.03. For inference, using VLLM [39] on an A100 GPU, SmolDocling achieves a page conversion time of 0.35 seconds while occupying only 0.489 GB of VRAM. The model supports a maximum sequence length of 8,192 tokens and is trained to process up to three pages at a time.

5.2 Quantitative Results

To enable valid cross-model comparisons with SmolDocling, we addressed multiple dataset and harmonization challenges:

- Public evaluation datasets lack comprehensive multi-task annotations with diverse layouts. We augmented DocLayNet [70] with text content and reading-order to enable evaluation of both layout analysis and text recognition. Table structure, code listing, formula and chart recognition are evaluated on specific datasets.
- Different models each follow their own conventions for output markup (e.g., Markdown, HTML, DocTags) and produce annotations which do not fully align in semantics. For example, SmolDocling is trained to produce 14 distinct layout-class tags via *DocTags*, while Qwen2.5-VL combines standard HTML tags and a few explicit class attributes with only partially compatible definitions. GOT and Nougat output Markdown, which can express paragraphs, lists and tables without location annotation.

- Input resolution of page images can affect model performance, but is rarely specified in literature. We standardized at 144 DPI (dots per inch), which reproduces enough detail to be perfectly legible by humans while keeping resource demand for inference reasonable.

Text Recognition (OCR). We first evaluate text recognition accuracy of SmolDocling, compared to Nougat, GOT and Qwen2.5-VL (Table 2), to verify accurate character transcription and reading order. The text-similarity metrics, adopted from [12] require plain formatting without non-textual elements. For different tasks, we conduct evaluations on diverse content types: full-page documents (from DocLayNet test set) in Markdown format, code snippets (from our SynthCodeNet dataset) in plaintext, and equations (from Im2Latex-230k [21]) in LaTeX format. For Qwen2.5-VL and SmolDocling, we convert HTML or DocTags output to Markdown while preserving formatting where possible.

Tables are omitted from the full-page output since they do not compare well through Markdown, due to loss of row and column spans. Code snippets must be reproduced with intact line breaks and indentation for accurate scoring. Formulas are evaluated separately in LaTeX format, which is natively produced by all the models tested in this work. The LaTeX outputs from all models, including the ground truth, undergo the same normalization procedure used for generating the SynthFormulaNet dataset. This ensures consistency and fairness in model comparisons.

SmolDocling significantly outperforms GOT, Nougat and also Qwen2.5-VL, which is 27 times larger, on every metric in full-page transcription (see Table 2. For the code parsing task, we report results exclusively for SmolDocling, as we introduce this task for the first time, and the other models were not explicitly trained for it. SmolDocling also demonstrates superior performance compared to Nougat and Qwen2.5-VL in the formula recognition task, closely matching GOT's results.

Layout. We evaluate element localization accuracy using the DocLayNet test set (Table 3), measuring the ability to reconstruct document representations that support content grounding and visual feature extraction. SmolDocling is compared against Qwen2.5-VL-7b [9], currently the only other VLM addressing this task. For a fair comparison, we map each model's label outputs and the ground-truth to six compatible classes: Text, Section Heading, List Item, Table, Picture, and Formula. Overall, SmolDocling outperforms Qwen2.5-VL-7b by a significant margin, however both models score far below a human baseline. This can be partially explained through inherent difficulty of the dataset, as indicated by relatively low human agreement scores. The scores are primarily impacted by moderate recall of element bounding boxes and different degrees of label confusion in both SmolDocling and Qwen2.5-VL. A majority of samples show accurate element bounding boxes as seen in the Appendix.

	SmolDocling	Qwen2.5-VL	Human
Formula	0.267	0.059	0.84
Table	0.266	0.262	0.79
List Item	0.296	0.090	0.87
Section Header	0.289	0.129	0.83
Text	0.204	0.180	0.85
Picture	0.066	0.078	0.70
Overall	0.231	0.133	0.82

Table 3: **Layout analysis comparison.** Evaluation of layout analysis task on DocLayNet with a subset of 6 class labels in mAP[0.5:0.95] metric. Additional reference points are provided for an estimate of human performance [70].

Table Structure Recognition. We evaluate the accuracy of recovering table structures, i.e. shaping columns and rows with the correct text content and spans, through the TEDS metric [104], once including table text content and once on structure only with ommitted text (see Table 4). For isolated assessment, we provide cropped table images from FinTabNet [103] and PubTables-1M [78] test sets, which have poor image quality (72 dpi, compression artifacts). SmolDocling performs competitively against significantly larger models, despite challenges with text transcription from the low-resolution image crops, a limitation likely attributable to insufficient training at such resolutions. Scores for structure only TEDS evaluation are therefore much stronger.

Chart Extraction. In this task, we evaluate the ability to convert cropped sections of charts into datapoint tables. All table ouptut is transformed into HTML and compared against the ground-truth HTML using TEDS score, as reported in Table 5. We attribute SmolDocling's performance on charts to the variability in ground-truth styles and the differing interpretations of representations across datasets, which we observe leads to inconsistencies in the model's predictions. Despite its significantly smaller size, SmolDocling achieves a competitive TEDs score.

Overall, SmolDocling achieves leading performance across a wide range of document conversion tasks and outperforms significantly larger models in text recognition, layout analysis and structure extraction. Furthermore, the model's

Method	Model Size	FinTabNet	PubTables-1M
SmolVLM [56]	2.2B	0.18	-
Granite Vision [81]	2.98B	0.54	-
TableFormer† [55]	52M	0.89	0.84
SmolDocling (Ours)	256M	0.52 (0.81)	0.65 (0.88)

Table 4: **Table structure and cell content reconstruction.** Numbers are given in TEDS metric including text content, with structure-only score in brackets where available. We produced results for SmolDocling and TableFormer, other results are reflected here as published. †TableFormer extracts the content through external OCR.

Model	Size	TEDs↑
Phi-3.5-vision	4B	0.40
Granite Vision [81]	3B	**0.95**
SmolVLM [56]	2.2B	0.02
Molmo-E	1B	0.54
SmolDocling (Ours)	256M	0.75

Table 5: **Chart analysis.** We compare SmolDocling to other small-sized VLMs on the chart-to-table task using the TEDS metric. The results in this table are based on the work reported in [25].

high performance in downstream tasks, such as code listing and formula recognition, demonstrates its versatility and real-world applicability. Notably, we establish benchmark results for the code listing recognition, setting a standard for future evaluations.

5.3 Qualitative Results

Several characteristics of SmolDocling output are not evident from task-based metrics alone. In this section, we highlight different output characteristics observed in practice. More illustrated samples can be found in the Appendix.

As presented in Figure 3, SmolDocling outputs a sequence of DocTags to encode content, structure and layout. A few notable traits beyond straight content reproduction include: 1) Labeling of page headers and footers, which allows ignoring these repetitive elements in downstream applications. 2) Linking of information, such as captions to tables or pictures, or list elements to parent list through nesting of tags. 3) Preserved line breaks and indentation for code snippets, while line wraps in paragraphs are correctly removed.

Among typical defects found in the output of SmolDocling are missing tags (e.g. absent location tags), malformed structure (such as missing closing tags), and endless repetition of tokens from some point in a page. Both of these phenomena are commonly seen in auto-regressive models and may prevent parsing or rendering the output correctly (see Appendix). Despite these limitations, SmolDocling demonstrates unique robustness advantages compared to ensemble systems like Docling, since conversion output is inferred in a single pass. In the latter, errors may accumulate throughout the ensemble model pipeline. For example, a mislocated table detection will lead to missing or distorted output of table structure and content, since this depends on correct table localization. In contrast, SmolDocling can reproduce the tables correctly, even with incorrectly determined location.

6 Conclusion

In this work, we introduce SmolDocling, an efficient and compact VLM optimized for document conversion while providing rich output representation. We also present a suite of new datasets with a unified format for document conversion, including the novel task of code listing transcription. We identify page element localization as a critical area requiring further refinement, where targeted techniques will significantly enhance performance in future iterations. Our results conclusively demonstrate that smaller models with unified, optimized output formats such as DocTags can effectively compete with significantly larger models, establishing a clear pathway for resource-efficient multi-task document understanding models.

References

[1] Grobid. https://github.com/kermitt2/grobid, 2008–2025.

[2] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When Smol Goes Big – Data-Centric Training of a Small Language Model, 2025.

[3] Amazon Web Services. Amazon Textract. https://aws.amazon.com/textract/, 2024. Accessed: 2024-11-11.

[4] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

[5] Anthropic. Claude-3.5. https://www.anthropic.com/news/claude-3-5-sonnet, 2024. Accessed: 2024-02-11.

[6] Christoph Auer, Michele Dolfi, André Carvalho, Cesar Berrospi Ramis, and Peter WJ Staar. Delivering Document Conversion as a Cloud Service with High Throughput and Responsiveness. In *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*, pages 363–373. IEEE, 2022.

[7] Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Nikolaos Livathinos, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Fabian Lindlbauer, Kasper Dinkla, et al. Docling Technical Report. *arXiv preprint arXiv:2408.09869*, 2024.

[8] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[9] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL Technical Report, 2025.

[10] Ali Furkan Biten, Rubèn Tito, Lluis Gomez, Ernest Valveny, and Dimosthenis Karatzas. Ocr-idl: Ocr annotations for industry document library dataset. In *European Conference on Computer Vision*, pages 241–252. Springer, 2022.

[11] Lukas Blecher. LaTeX-OCR, 2024. Accessed: 2024-11-09.

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

[13] Jinyue Chen, Lingyu Kong, Haoran Wei, Chenglong Liu, Zheng Ge, Liang Zhao, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. OneChart: Purify the Chart Structural Extraction via One Auxiliary Token. *arXiv preprint arXiv:2404.09987*, 2024.

[14] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[15] Pyecharts Contributors. pyecharts: Python library for Echarts, 2024. Accessed: 2024-11-06.

[16] Pygments Contributors. Pygments: Python syntax highlighter, 2024. Accessed: 2024-11-06.

[17] Brian Davis, Bryan Morse, Brian Price, Chris Tensmeyer, Curtis Wigington, and Vlad Morariu. End-to-end document recognition and understanding with dessurt. In *European Conference on Computer Vision*, pages 280–296. Springer, 2022.

[18] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M Rush. Image-to-markup generation with coarse-to-fine attention. In *International Conference on Machine Learning*, pages 980–989. PMLR, 2017.

[19] Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Bin Wang, Linke Ouyang, Xilin Wei, Songyang Zhang, Haodong Duan, Maosong Cao, Wenwei Zhang, Yining Li, Hang Yan, Yang Gao, Xinyue Zhang, Wei Li, Jingwen Li, Kai Chen, Conghui He, Xingcheng Zhang, Yu Qiao, Dahua Lin, and Jiaqi Wang. InternLM-XComposer2: Mastering Free-form Text-Image Composition and Comprehension in Vision-Language Large Model. *arXiv preprint arXiv:2401.16420*, 2024.

[20] Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Bin Wang, Linke Ouyang, Songyang Zhang, Haodong Duan, Wenwei Zhang, Yining Li, Hang Yan, Yang Gao, Zhe Chen, Xinyue Zhang, Wei Li, Jingwen Li, Wenhai Wang, Kai Chen, Conghui He, Xingcheng Zhang, Jifeng Dai, Yu Qiao, Dahua Lin, and Jiaqi Wang. InternLM-XComposer2-4KHD: A Pioneering Large Vision-Language Model Handling Resolutions from 336 Pixels to 4K HD. *arXiv preprint arXiv:2404.06512*, 2024.

[21] Gregory Eritsyan. Im2LaTeX 230K. https://www.kaggle.com/datasets/gregoryeritsyan/im2latex-230k, 2023. Accessed: 2025-03-06.

[22] Hugging Face. latex-formulas Dataset, 2024. Accessed: 2024-11-09.

[23] Philippe Gervais, Asya Fadeeva, and Andrii Maksai. Mathwriting: A dataset for handwritten mathematical expression recognition. *URL https://arxiv. org/abs/2404.10690*, 2024.

[24] Google Cloud. Document AI. https://cloud.google.com/document-ai, 2024. Accessed: 2024-11-11.

[25] IBM Granite Team. Granite 3.0 Language Models, 2024.

[26] Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. Chartllama: A multimodal llm for chart understanding and generation. *arXiv preprint arXiv:2311.16483*, 2023.

[27] Anwen Hu, Haiyang Xu, Jiabo Ye, Ming Yan, Liang Zhang, Bo Zhang, Chen Li, Ji Zhang, Qin Jin, Fei Huang, et al. mplug-docowl 1.5: Unified structure learning for ocr-free document understanding. *arXiv preprint arXiv:2403.12895*, 2024.

[28] Anwen Hu, Haiyang Xu, Liang Zhang, Jiabo Ye, Ming Yan, Ji Zhang, Qin Jin, Fei Huang, and Jingren Zhou. mplug-docowl2: High-resolution compressing for ocr-free multi-page document understanding. *arXiv preprint arXiv:2409.03420*, 2024.

[29] Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4083–4091, 2022.

[30] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[31] Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. Funsd: A dataset for form understanding in noisy scanned documents. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 2, pages 1–6. IEEE, 2019.

[32] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[33] Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. Figureqa: An annotated figure dataset for visual reasoning. *arXiv preprint arXiv:1710.07300*, 2017.

[34] Shankar Kantharaj, Rixie Tiffany Ko Leong, Xiang Lin, Ahmed Masry, Megh Thakkar, Enamul Hoque, and Shafiq Joty. Chart-to-text: A large-scale benchmark for chart summarization. *arXiv preprint arXiv:2203.06486*, 2022.

[35] Geewook Kim, Teakgyu Hong, Moonbin Yim, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. Donut: Document understanding transformer without ocr. *arXiv preprint arXiv:2111.15664*, 7(15):2, 2021.

[36] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, Leonid Zaslavsky, Jian Zhang, and Evan E Bolton. PubChem 2019 update: improved access to chemical data. *Nucleic Acids Research*, 47(D1):D1102–D1109, 10 2018.

[37] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[38] Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. The Stack: 3 TB of permissively licensed source code. *Preprint*, 2022.

[39] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[40] Greg Landrum. RDKit: Open-Source Cheminformatics Software. http://www.rdkit.org/. Accessed: 1 January 2023.

[41] Hugo Laurençon, Andrés Marafioti, Victor Sanh, and Léo Tronchon. Building and better understanding vision-language models: insights and future directions., 2024.

[42] Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. What matters when building vision-language models?, 2024.

[43] 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 *International Conference on Machine Learning*, pages 18893–18912. PMLR, 2023.

[44] David Lewis, Gady Agam, Shlomo Argamon, Ophir Frieder, David Grossman, and Jefferson Heard. Building a test collection for complex document information processing. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 665–666, 2006.

[45] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

[46] Feng Li, Renrui Zhang, Hao Zhang, Yuanhan Zhang, Bo Li, Wei Li, Zejun Ma, and Chunyuan Li. Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models. *arXiv preprint arXiv:2407.07895*, 2024.

[47] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[48] Han Lin, Peng Yang, and Fanlong Zhang. Review of scene text detection and recognition. *Archives of computational methods in engineering*, 27(2):433–454, April 2020.

[49] Fangyu Liu, Julian Martin Eisenschlos, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Wenhu Chen, Nigel Collier, and Yasemin Altun. Deplot: One-shot visual language reasoning by plot-to-table translation. *arXiv preprint arXiv:2212.10505*, 2022.

[50] Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier, and Julian Martin Eisenschlos. Matcha: Enhancing visual language pretraining with math reasoning and chart derendering. *arXiv preprint arXiv:2212.09662*, 2022.

[51] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved Baselines with Visual Instruction Tuning, 2023.

[52] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. In *NeurIPS*, 2023.

[53] Nikolaos Livathinos, Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Kasper Dinkla, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. Docling: An Efficient Open-Source Toolkit for AI-driven Document Conversion, 2025.

[54] Nam Ly., Atsuhiro Takasu., Phuc Nguyen., and Hideaki Takeda. Rethinking Image-Based Table Recognition Using Weakly Supervised Methods. pages 872–880, 2023.

[55] Maksym Lysak, Ahmed Nassar, Nikolaos Livathinos, Christoph Auer, and Peter Staar. Optimized Table Tokenization for Table Structure Recognition. In Gernot A. Fink, Rajiv Jain, Koichi Kise, and Richard Zanibbi, editors, *Document Analysis and Recognition - ICDAR 2023*, pages 37–50, Cham, 2023. Springer Nature Switzerland.

[56] Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Vaibhav Srivastav, Joshua Lochner, Hugo Larcher, Mathieu Morlon, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. SmolVLM: Redefining small and efficient multimodal models. 2025.

[57] Ahmed Masry, Parsa Kavehzadeh, Xuan Long Do, Enamul Hoque, and Shafiq Joty. Unichart: A universal vision-language pretrained model for chart comprehension and reasoning. *arXiv preprint arXiv:2305.14761*, 2023.

[58] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*, 2022.

[59] Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209, 2021.

[60] Fanqing Meng, Wenqi Shao, Quanfeng Lu, Peng Gao, Kaipeng Zhang, Yu Qiao, and Ping Luo. Chartassisstant: A universal chart multimodal language model via chart-to-table pre-training and multitask instruction tuning. *arXiv preprint arXiv:2401.02384*, 2024.

[61] Nitesh Methani, Pritha Ganguly, Mitesh M. Khapra, and Pratyush Kumar. PlotQA: Reasoning over Scientific Plots. In *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[62] Microsoft Azure. AI Document Intelligence. `https://azure.microsoft.com/en-us/products/ai-services/ai-document-intelligence`, 2024. Accessed: 2024-11-11.

[63] Lucas Morin, Martin Danelljan, Maria Isabel Agea, Ahmed Nassar, Valery Weber, Ingmar Meijer, Peter Staar, and Fisher Yu. MolGrapher: Graph-based Visual Recognition of Chemical Structures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19552–19561, October 2023.

[64] Lucas Morin, Valéry Weber, Gerhard Ingmar Meijer, Fisher Yu, and Peter W. J. Staar. PatCID: an open-access dataset of chemical structures in patent documents. *Nature Communications*, 15(1):6532, Aug 2024.

[65] OpenAI. Hello GPT-4O. `https://openai.com/index/hello-gpt-4o/`, 2024. Accessed: 2024-02-09, 2024-02-11, 2024-02-12.

[66] R OpenAI. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2(5), 2023.

[67] Vik Paruchuri. Marker: Convert PDF to Markdown Quickly with High Accuracy. https://github.com/VikParuchuri/marker, 2024.

[68] Vik Paruchuri. Texify. `https://github.com/VikParuchuri/texify`, 2024. Accessed: 2024-11-11.

[69] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. *arXiv preprint arXiv:2306.14824*, 2023.

[70] Birgit Pfitzmann, Christoph Auer, Michele Dolfi, Ahmed S. Nassar, and Peter 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*, KDD '22, page 3743–3751, New York, NY, USA, 2022. Association for Computing Machinery.

[71] Ruchir Puri, David Kung, Geert Janssen, Wei Zhang, Giacomo Domeniconi, Vladmir Zolotov, Julian Dolby, Jie Chen, Mihir Choudhury, Lindsey Decker, Veronika Thost, Luca Buratti, Saurabh Pujar, Shyam Ramji, Ulrich Finkler, Susan Malaika, and Frederick Reiss. CodeNet: A Large-Scale AI for Code Dataset for Learning a Diversity of Coding Tasks, 2021.

[72] Edward O. Pyzer-Knapp, Matteo Manica, Peter Staar, Lucas Morin, Patrick Ruch, Teodoro Laino, John R. Smith, and Alessandro Curioni. Foundation models for materials discovery – current state and future directions. *npj Computational Materials*, 11(1):61, Mar 2025.

[73] Yujie Qian, Jiang Guo, Zhengkai Tu, Zhening Li, Connor W. Coley, and Regina Barzilay. MolScribe: Robust Molecular Structure Recognition with Image-to-Graph Generation. *Journal of Chemical Information and Modeling*, 63(7):1925–1934, Apr 2023.

[74] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[75] Kohulan Rajan, Henning Otto Brinkhaus, M. Isabel Agea, Achim Zielesny, and Christoph Steinbeck. DECIMER.ai: an open platform for automated optical chemical structure identification, segmentation and recognition in scientific publications. *Nature Communications*, 14(1):5045, Aug 2023.

[76] Johannes Rausch, Octavio Martinez, Fabian Bissig, Ce Zhang, and Stefan Feuerriegel. Docparser: Hierarchical document structure parsing from renderings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4328–4338, 2021.

[77] Felix M Schmitt-Koopmann, Elaine M Huang, Hans-Peter Hutter, Thilo Stadelmann, and Alireza Darvishy. MathNet: A Data-Centric Approach for Printed Mathematical Expression Recognition. *IEEE Access*, 2024.

[78] Brandon Smock, Rohith Pesala, and Robin Abraham. PubTables-1M: Towards comprehensive table extraction from unstructured documents. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4624–4632, 2022.

[79] Zineng Tang, Ziyi Yang, Guoxin Wang, Yuwei Fang, Yang Liu, Chenguang Zhu, Michael Zeng, Cha Zhang, and Mohit Bansal. Unifying vision, text, and layout for universal document processing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19254–19264, 2023.

[80] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[81] Granite Vision Team, Leonid Karlinsky, Assaf Arbelle, Abraham Daniels, Ahmed Nassar, Amit Alfassi, Bo Wu, Eli Schwartz, Dhiraj Joshi, Jovana Kondic, Nimrod Shabtay, Pengyuan Li, Roei Herzig, Shafiq Abedin, Shaked Perek, Sivan Harary, Udi Barzelay, Adi Raz Goldfarb, Aude Oliva, Ben Wieles, Bishwaranjan Bhattacharjee, Brandon Huang, Christoph Auer, Dan Gutfreund, David Beymer, David Wood, Hilde Kuehne, Jacob Hansen, Joseph Shtok, Ken Wong, Luis Angel Bathen, Mayank Mishra, Maksym Lysak, Michele Dolfi, Mikhail Yurochkin, Nikolaos Livathinos, Nimrod Harel, Ophir Azulai, Oshri Naparstek, Rafael Teixeira de Lima, Rameswar Panda, Sivan Doveh, Shubham Gupta, Subhro Das, Syed Zawad, Yusik Kim, Zexue He, Alexander Brooks, Gabe Goodhart, Anita Govindjee, Derek Leist, Ibrahim Ibrahim, Aya Soffer, David Cox, Kate Soule, Luis Lastras, Nirmit Desai, Shila Ofek-koifman, Sriram Raghavan, Tanveer Syeda-Mahmood, Peter Staar, Tal Drory, and Rogerio Feris. Granite Vision: a lightweight, open-source multimodal model for enterprise Intelligence, 2025.

[82] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[83] Michał Turski, Tomasz Stanisławek, Karol Kaczmarek, Paweł Dyda, and Filip Graliński. CCpdf: Building a High Quality Corpus for Visually Rich Documents from Web Crawl Data. In Gernot A. Fink, Rajiv Jain, Koichi Kise, and Richard Zanibbi, editors, *Document Analysis and Recognition - ICDAR 2023*, pages 348–365, Cham, 2023. Springer Nature Switzerland.

[84] Unstructured.io Team. Unstructured.io: Open-Source Pre-Processing Tools for Unstructured Data. https://unstructured.io, 2024. Accessed: 2024-11-19.

[85] Vicuna. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality. `https://vicuna.lmsys.org/`, 2023.

[86] Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, Bo Zhang, Liqun Wei, Zhihao Sui, Wei Li, Botian Shi, Yu Qiao, Dahua Lin, and Conghui He. MinerU: An Open-Source Solution for Precise Document Content Extraction, 2024.

[87] Michael L Waskom. Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

[88] Maurice Weber, Carlo Siebenschuh, Rory Butler, Anton Alexandrov, Valdemar Thanner, Georgios Tsolakis, Haris Jabbar, Ian Foster, Bo Li, Rick Stevens, et al. WordScape: a Pipeline to extract multilingual, visually rich Documents with Layout Annotations from Web Crawl Data. *Advances in Neural Information Processing Systems*, 36, 2024.

[89] Haoran Wei, Chenglong Liu, Jinyue Chen, Jia Wang, Lingyu Kong, Yanming Xu, Zheng Ge, Liang Zhao, Jianjian Sun, Yuang Peng, et al. General OCR Theory: Towards OCR-2.0 via a Unified End-to-end Model. *arXiv preprint arXiv:2409.01704*, 2024.

[90] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[91] David Weininger. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, Feb 1988.

[92] Wes McKinney and the Pandas Development Team. Pandas: powerful Python data analysis toolkit, Release 1.4.4. `https://pandas.pydata.org/pandas-docs/version/1.4/pandas.pdf`, 2022.

[93] Renqiu Xia, Bo Zhang, Haoyang Peng, Hancheng Ye, Xiangchao Yan, Peng Ye, Botian Shi, Junchi Yan, and Yu Qiao. StructChart: Perception, Structuring, Reasoning for Visual Chart Understanding. *arXiv preprint arXiv:2309.11268*, 2023.

[94] Renqiu Xia, Bo Zhang, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Min Dou, Botian Shi, Junchi Yan, et al. Chartx & chartvlm: A versatile benchmark and foundation model for complicated chart reasoning. *arXiv preprint arXiv:2402.12185*, 2024.

[95] Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, et al. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. *arXiv preprint arXiv:2012.14740*, 2020.

[96] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1192–1200, 2020.

[97] Zuoyu Yan, Xiaode Zhang, Liangcai Gao, Ke Yuan, and Zhi Tang. ConvMath: a convolutional sequence network for mathematical expression recognition. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4566–4572. IEEE, 2021.

[98] Jiabo Ye, Anwen Hu, Haiyang Xu, Qinghao Ye, Ming Yan, Guohai Xu, Chenliang Li, Junfeng Tian, Qi Qian, Ji Zhang, et al. Ureader: Universal ocr-free visually-situated language understanding with multimodal large language model. *arXiv preprint arXiv:2310.05126*, 2023.

[99] Pan Zhang, Xiaoyi Dong, Bin Wang, Yuhang Cao, Chao Xu, Linke Ouyang, Zhiyuan Zhao, Shuangrui Ding, Songyang Zhang, Haodong Duan, Wenwei Zhang, Hang Yan, Xinyue Zhang, Wei Li, Jingwen Li, Kai Chen, Conghui He, Xingcheng Zhang, Yu Qiao, Dahua Lin, and Jiaqi Wang. InternLM-XComposer: A Vision-Language Large Model for Advanced Text-image Comprehension and Composition. *arXiv preprint arXiv:2309.15112*, 2023.

[100] Pan Zhang, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Rui Qian, Lin Chen, Qipeng Guo, Haodong Duan, Bin Wang, Linke Ouyang, Songyang Zhang, Wenwei Zhang, Yining Li, Yang Gao, Peng Sun, Xinyue Zhang, Wei Li, Jingwen Li, Wenhai Wang, Hang Yan, Conghui He, Xingcheng Zhang, Kai Chen, Jifeng Dai, Yu Qiao, Dahua Lin, and Jiaqi Wang. InternLM-XComposer-2.5: A Versatile Large Vision Language Model Supporting Long-Contextual Input and Output. *arXiv preprint arXiv:2407.03320*, 2024.

[101] Qiming Zhang, Jing Zhang, Yufei Xu, and Dacheng Tao. Vision Transformer with Quadrangle Attention. *arXiv preprint arXiv:2303.15105*, 2023.

[102] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *URL https://arxiv. org/abs/2205.01068*, 3:19–0, 2023.

[103] Xinyi Zheng, Douglas Burdick, Lucian Popa, Xu Zhong, and Nancy Xin Ru Wang. Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 697–706, 2021.

[104] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. Image-based table recognition: data, model, and evaluation. *arXiv preprint arXiv:1911.10683*, 2019.

[105] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

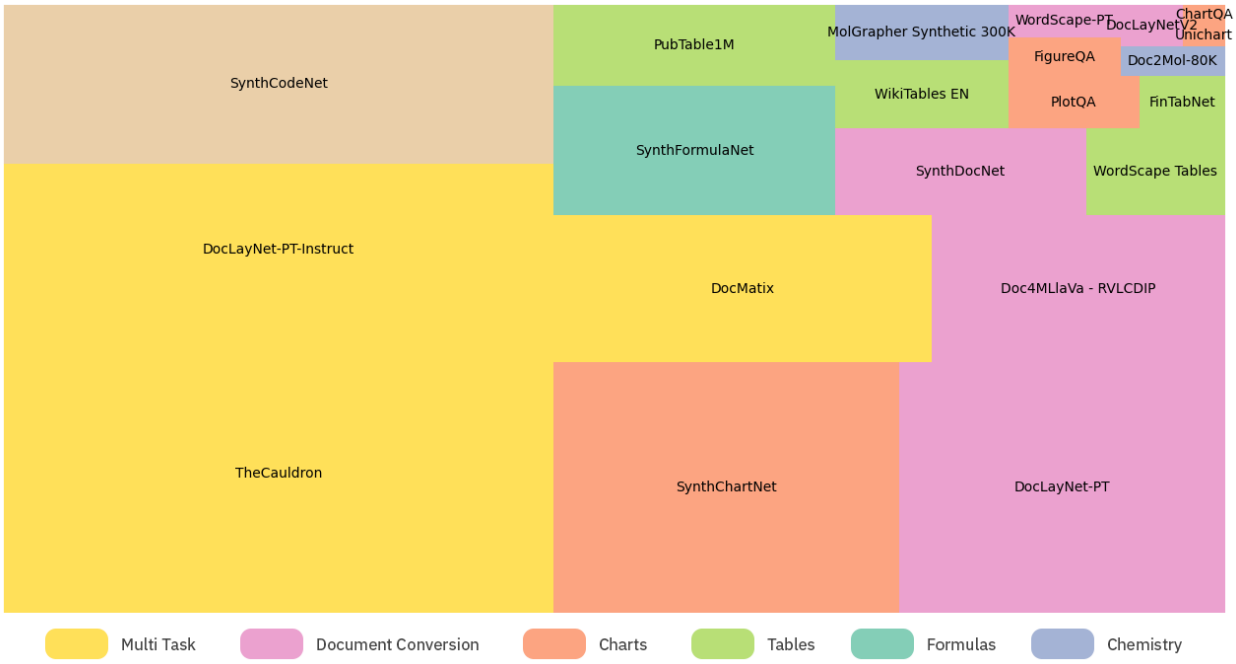


Figure 3: **Training datasets.** A treemap visualization of dataset sizes contributing to the training of SmolDocling, categorized by type. Each rectangle represents a dataset, with its area proportional to the total dataset size. Each color indicates a dataset type. *"Multi-Task"* refers to datasets that include multiple different tasks.

A Additional Datasets Details

A.1 SynthChartNet

SynthChartNet includes four distinct types of charts: line, pie, bar, and stacked bar, generated using three visualization libraries: Matplotlib [30], Seaborn [87], and Pyecharts [15]. Each data set was rendered through all three libraries, producing a final dataset of 2.5 million visually diverse charts. The rendering process was designed to leverage the full range of features available in these libraries, ensuring visual variety while maintaining the plausibility of the data presented. Figure 4 depicts 10 samples from SynthChartNet.

A.2 SynthCodeNet

To render the code snippets of SynthCodeNet, we employed two rendering libraries: LaTeX and Pygments [16]. LaTex, in conjunction with its listings package, enabled the generation of code renderings similar to those commonly seen in academic and technical publications, while Pygments facilitated renderings with aesthetics resembling those of IDEs. The rendering process utilized the complete set of features provided by these libraries to enhance visual diversity while preserving the plausibility of the presented data. This included randomizing various stylistic parameters such as font, font size, style, weight, line spacing, tab size, background color, line numbering (including randomizing the starting line number), and programming language-specific syntax highlighting. Figure 5 shows 4 samples from SynthCodeNet.

A.3 SynthFormulaNet

The extracted formulas of SynthFormulaNet underwent a rigorous normalization procedure to ensure the model was trained only on correct and standardized code. The normalization process consisted of the following major steps:

1. **Token Filtering and Removal**: Unnecessary or redundant LaTeX tokens were removed or replaced based on predefined policies. This included:
   - Eliminating unwanted tokens.
   - Replacing tokens with equivalent ones for consistency.

Figure 4: **Samples from SynthChartNet.**

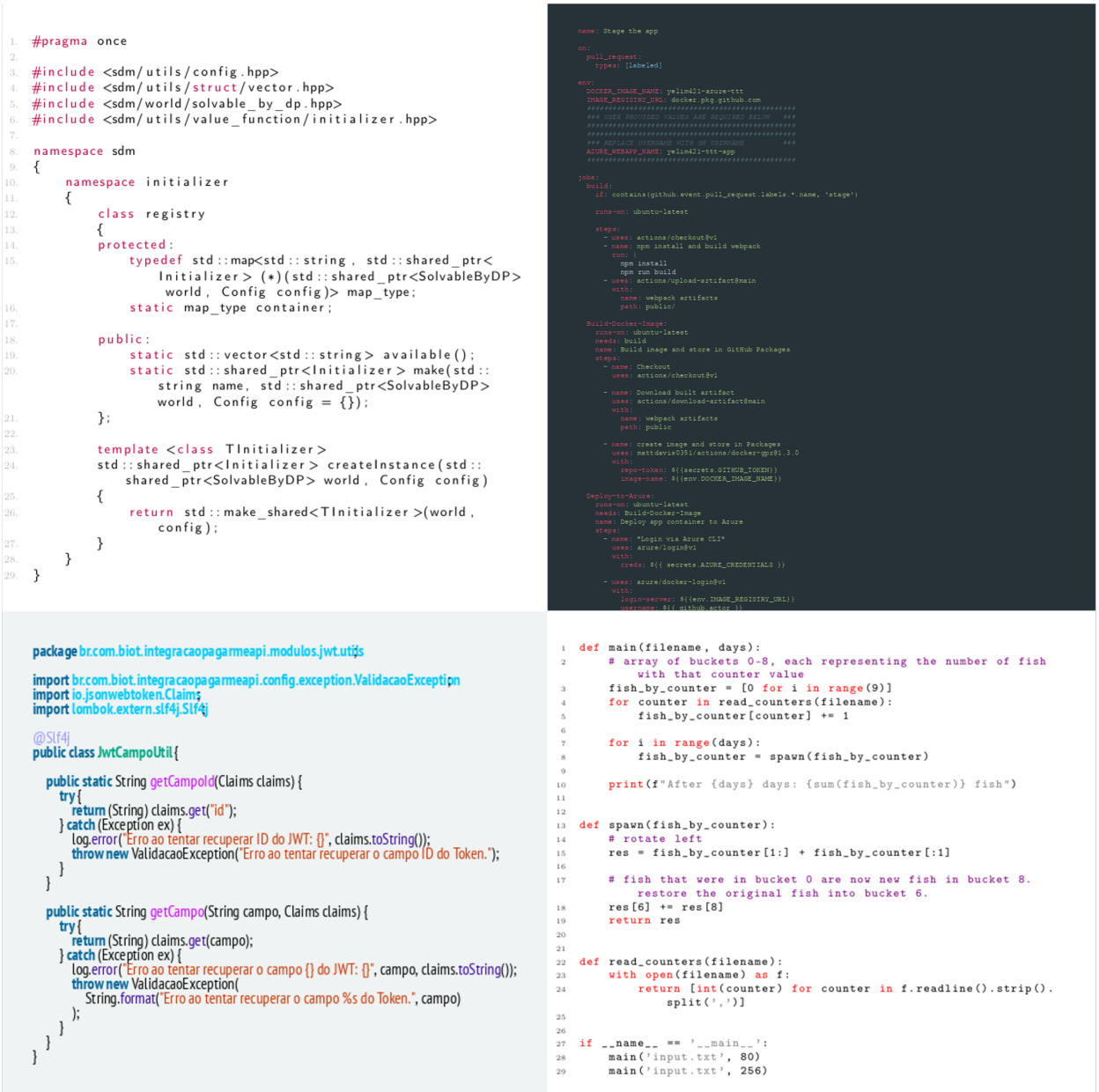


Figure 5: **Samples from SynthCodeNet.**

$$\begin{aligned}\sum_{k\in J\cap K_m} e^{-i\varphi_k} m_{2Q_k}(e^{i\varphi_k} f)\nabla\chi_k = &\sum_{k\in J\cap K_m} \left(e^{-i\varphi_k} m_{2Q_k}(e^{i\varphi_k} f) - e^{-i\tilde{\varphi}_m} m_{2Q_k}(e^{i\tilde{\varphi}_m} f)\right)\nabla\chi_k \\ &+ \sum_{k\in J\cap K_m} e^{-i\tilde{\varphi}_m}\left(m_{2Q_k}(e^{i\tilde{\varphi}_m} f) - m_{\tilde{Q}_m}(e^{i\tilde{\varphi}_m} f)\right)\nabla\chi_k \\ &+ \sum_{k\in J\cap K_m} e^{-i\tilde{\varphi}_m} m_{\tilde{Q}_m}(e^{i\tilde{\varphi}_m} f)\nabla\chi_k \\ &= I + II + III.\end{aligned} \tag{Y}$$

$$Rd \le \alpha \ln\left(\frac{(A_y+\lambda_y)^{A_y+\lambda_y}}{(A_z+\lambda_z)^{A_z+\lambda_z}}\right) + (1-\alpha)\ln\left(\frac{\lambda_y^{\lambda_y}}{\lambda_z^{\lambda_z}}\right) - \ln\left(\frac{(A_y\alpha+\lambda_y)^{A_y\alpha+\lambda_y}}{(A_z\alpha+\lambda_z)^{A_z\alpha+\lambda_z}}\right) \tag{4.30}$$

$$R \le \alpha \ln\left((A_y+\lambda_y)^{A_y+\lambda_y}\right) + (1-\alpha)\ln\left(\lambda_y^{\lambda_y}\right) - \ln\left((A_y\alpha+\lambda_y)^{A_y\alpha+\lambda_y}\right) \tag{4.31}$$

$$d \le 1 \tag{4.32}$$

$$u_{ij}^{(0)} = \begin{cases} \frac{1}{(1+q)^{j-1}}, & (i=1), \\ \frac{q}{(1+q)^{j-i+1}}, & (2\le i\le j), \\ 0, & (j+1\le i\le N). \end{cases} \tag{L}$$

$$d\Phi = \left(\frac{2\pi}{k}\right)^2 s_z^2 \tilde{a}(-\rho s)\tilde{a}^*(-\rho s)\, d\Omega.$$

$$\begin{aligned}
&\mathscr{L}_1 = \left\{ \max(\frac{\bar{a}}{\bar{b}}\gamma, \frac{\gamma(\bar{a}+b)-\sqrt{\gamma^2(\bar{a}+b)^2-4a\bar{b}}}{2\bar{b}}) \le \beta \le \frac{\gamma(\bar{a}+b)+\sqrt{\gamma^2(\bar{a}+b)^2-4a\bar{b}}}{2\bar{b}}. \right\} \\
&\mathscr{L}_2 = \left\{ \frac{(a+\bar{b})+\sqrt{(a+\bar{b})^2-4\bar{a}b\gamma^2}}{2b\gamma} \le \beta \le \frac{\bar{a}}{\bar{b}}\gamma \right\} \\
&\mathscr{L}_3 = \left\{ \beta \le \min(\frac{\bar{a}}{\bar{b}}\gamma, \frac{(a+\bar{b})-\sqrt{(a+\bar{b})^2-4\bar{a}b\gamma^2}}{2b\gamma}) \right\} \\
&\mathscr{L}_4 = \left\{ \beta \le \min(\frac{b\gamma}{a}, \frac{\gamma(\bar{a}+b)-\sqrt{\gamma^2(\bar{a}+b)^2-4a\bar{b}}}{2a}) \right\} \\
&\mathscr{L}_5 = \left\{ \frac{\gamma(\bar{a}+b)+\sqrt{\gamma^2(\bar{a}+b)^2-4a\bar{b}}}{2a} \le \beta \le \frac{b\gamma}{a} \right\} \\
&\mathscr{L}_6 = \left\{ \max(\frac{b\gamma}{a}, \frac{(a+\bar{b})-\sqrt{(a+\bar{b})^2-4\bar{a}b\gamma^2}}{2\bar{a}\gamma}) \le \beta \le \frac{(a+\bar{b})+\sqrt{(a+\bar{b})^2-4\bar{a}b\gamma^2}}{2\bar{a}\gamma} \right\},
\end{aligned} \tag{x}$$

$$\begin{aligned}\int_{S^{n-1}} v(s)\, d\mu^\sigma \ge &\frac{(n-2)\omega_{n-1}}{2} m^{U^\kappa b} \\ &+ \int_s^\infty \left(1 - \frac{\cosh s}{\cosh r}\frac{u_0(r)}{u_0(s)}\right)\cosh r \sinh^{n-1} r \left(\int_{S^{n-1}} f(u(r,\theta))\, d\mu^\sigma\right) dr\end{aligned} \tag{21.64}$$

Figure 6: **Samples from SynthFormulaNet.**

2. **Structural Normalization**: The equation structure was standardized by:
   - Normalizing spacing between tokens.
   - Standardizing left and right delimiters.
   - Ensuring consistent use of tokens that require one or two braces.
3. **Token Simplification**: Redundant tokens and patterns were simplified, including:
   - Collapsing consecutive redundant tokens (e.g., repeated primes, dots, or excessive spacing).
   - Converting special constructs such as `\Big` into `\left` and `\right` for consistency.

The extracted equations were subsequently rendered using LaTex at 120 dpi. Several aspects of the rendering procedure have been randomized to ensure the visual diversity of the dataset. Each formula was rendered using a randomly selected font from a set of 85 possible options, with variations in font size, style, weight, and line spacing. Additionally, with a certain probability, each formula was enclosed within an "align" environment, allowing for the inclusion of a randomly generated equation number on either the left or right side of the equation. The equation number itself was chosen at random from a diverse set of possibilities, including small and large numbers, integers, floating-point values, or letters. Furthermore, it could be enclosed in single or double parentheses, brackets, or curly braces and was sometimes preceded by text labels such as “Eqn.", “Eqn", “Eq.", or “Equation". Figure 6 depicts 6 samples from SynthFormulaNet.

A.4 DocTags vocabulary

Here we provide a reference to a complete vocabulary of *DocTags* for *SmolDocling*, and their meaning.

We represent tags in an XML-like style of notation. Some tags have an opening and closing part, and wrap around textual content (or other tags) for example: `<text>hello world</text>`. Other tags are standalone and do not have a closing tag; they mark a special instruction, for example: `<page_break>`

The complete *DocTags* snippet can represent a page or multiple pages, wrapped into `<doctag>...</doctag>`. When content represents multiple pages, we use `<page_break>` tag as a separator.

Within the scope of `<doctag>` we place high-level, tags that wrap around the textual content of certain document blocks and identify the type of the block. These tags are: `<text>`, `<caption>`, `<footnote>`, `<formula>`, `<title>`, `<page_footer>`, `<page_header>`, `<picture>`, `<section_header>`, `<document_index>`, `<code>`, `<otsl>`, `<list_item>`, `<ordered_list>`, `<unordered_list>`

Each element can nest additional standalone location tags that encode its position on the page as a bounding box, represented in DocTags as `<loc_x1><loc_y1><loc_x2><loc_y2>`. Every x,y pair of coordinates belongs to a fixed grid. In our case we used integer values in the range from 0 to 500 which proportionally maps to width and height of the page.

`<otsl>` element contains table representation with following OTSL tags, that are standalone and used to describe structure of a table: `<fcel>`, `<ecel>`, `<lcel>`, `<ucel>`, `<xcel>`, `<nl>`. These tags follow rules as described in [55], we additionally differentiate OTSL notion of a cell into `<fcel>` - full cell, or cell which contain content, and `<ecel>` - empty cell. We also augment tabular structure information with extra tags `<ched>`, `<rhed>`, `<srow>` that replace `<fcel>` to describe cells which belong to column and row headers of the table, as well as table section accordingly. We mark such headers when ground truth about table headers from the underlying dataset is available.

`<list_item>` elements are placed within `<ordered_list>` or `<unordered_list>` and define weather it’s enumerated (ordered) list or not.

`<picture>` and `<otsl>` elements by themselves can encapsulate `<caption>` tag if appropriate picture or table has it’s own caption. This way we can connect extra descriptive information to illustrations and tables in the document.

`<code>` elements contain pieces of code as a content, and as such, respect tabulation and line breaks. Code elements also contains special standalone classification tag - `<_programming-language_>` where *programming-language* is an appropriate programming language name. Supported values (57 in total) for programming language names are: *Ada*, *Awk*, *Bash*, *bc*, *C*, *C#*, *C++*, *CMake*, *COBOL*, *CSS*, *Ceylon*, *Clojure*, *Crystal*, *Cuda*, *Cython*, *D*, *Dart*, *dc*, *Dockerfile*, *Elixir*, *Erlang*, *FORTRAN*, *Forth*, *Go*, *HTML*, *Haskell*, *Haxe*, *Java*, *JavaScript*, *Julia*, *Kotlin*, *Lisp*, *Lua*, *Matlab*, *MoonScript*, *Nim*, *OCaml*, *ObjectiveC*, *Octave*, *PHP*, *Pascal*, *Perl*, *Prolog*, *Python*, *Racket*, *Ruby*, *Rust*, *SML*, *SQL*, *Scala*, *Scheme*, *Swift*, *TypeScript*, *unknown*, *VisualBasic*, *XML*, *YAML*

Within `<picture>` elements we include picture classification with extra standalone tags `<image_class>`. We classify images from our datasets into following categories: *natural_image*, *pie_chart*, *bar_chart*, *line_chart*,

flow_chart, *scatter_chart*, *heatmap*, *remote_sensing*, *chemistry_molecular_structure*, *chemistry_markush_structure*, *icon*, *logo*, *signature*, *stamp*, *qr_code*, *bar_code*, *screenshot*, *map*, *stratigraphic_chart*, *cad_drawing*, *electrical_diagram*

B Additional Details about the SmolDocling

Instruction	Description
Full conversion	Convert this page to docling.
Chart	Convert chart to table (e.g., <chart>).
Formula	Convert formula to LaTeX (e.g., <formula>).
Code	Convert code to text (e.g., <code>).
Table	Convert table to OTSL (e.g., <otsl>).
No-Code Actions/Pipelines	OCR the text in a specific location: <loc_155><loc_233><loc_206><loc_237>
	Identify element at: <loc_247><loc_482><loc_252><loc_486>
	Find all 'text' elements on the page, retrieve all section headers.
	Detect footer elements on the page.

Table 6: **Supported Conversion Instructions.**

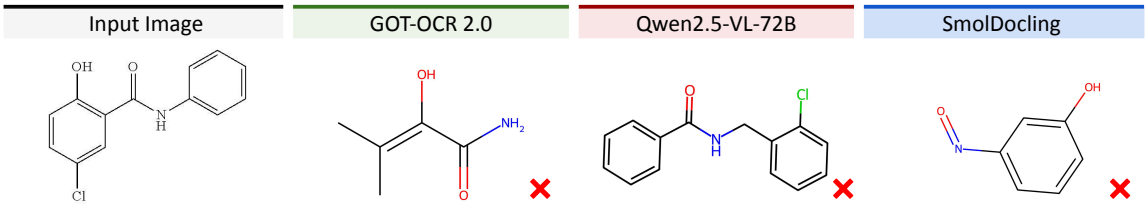


Figure 7: **Qualitative comparison for molecule image recognition.** Predictions are shown for GOT-OCR 2.0, Qwen2.5-VL-72B and SmolDocling on a simple molecule image.

C Additional Qualitative Analysis

C.1 Molecule Image Recognition Experiment

Extracting molecule images from documents has potential to accelerate research in chemistry and materials discovery [72]. Some recent document understanding models, such as GOT-OCR 2.0 [89] and Qwen2.5-VL [9], claim to perform molecule image recognition in documents. Following this direction, we experimented with training SmolDocling for molecule image recognition on full document pages and cropped images. In order to predict molecule structures (graphs) with SmolDocling, we use the molecule string identifier named SMILES [91].

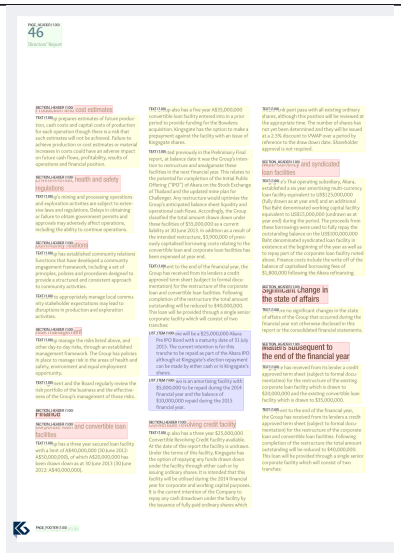
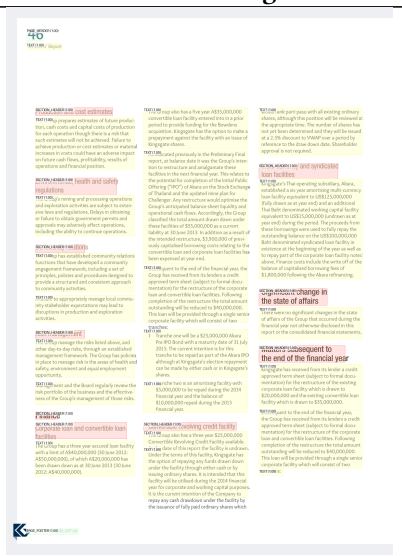
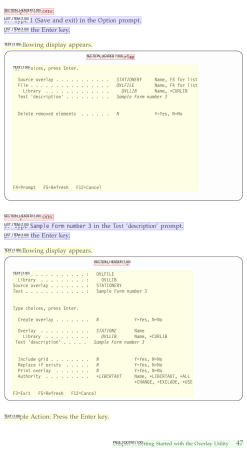
Datasets. For training on cropped images, we use the MolGrapher-Synthetic-300K dataset [63]. The dataset contains 300K synthetic samples. It is created using real chemical-structures retrieved from the database PubChem [36] which are then synthetically drawn using the library RDKit [40]. For training of full pages, we use Doc2Mol-80K, a subset of the PatCID dataset [64]. Our subset contains 80 000 real patent documents published in the United-States between 2022 and 2024.

Evaluation. Figure 7 shows an example of predicted molecules for a simple molecule image. While some models capture high-level molecular features, they consistently fail to reconstruct its detailed structure. Notably, this molecule is trivial for specialized models such as MolGrapher [63], MolScribe [73] or DECIMER [75], which accurately recognize its structure. Reducing the gap between general document understanding models and specialized models may require incorporating a specialized vocabulary for SMILES tokens, explicitly encoding atoms and bonds, and extending the SMILES sequence with localization tokens for each atom.

C.2 Layout analysis samples

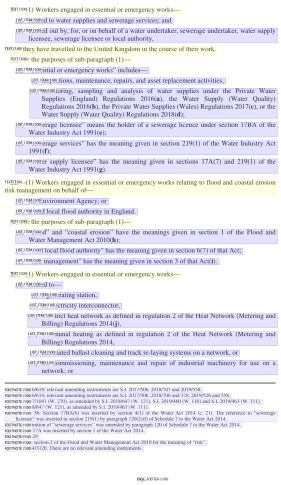
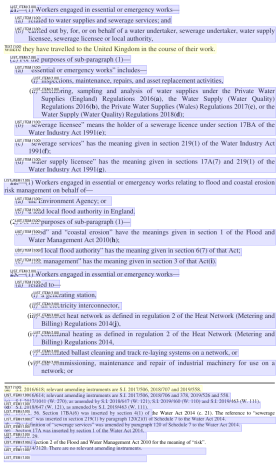
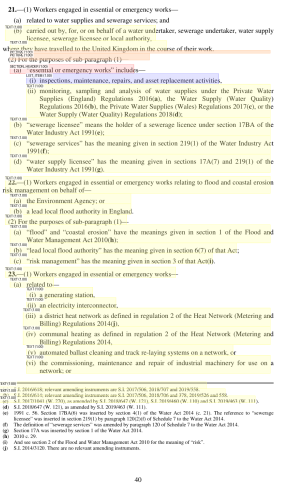
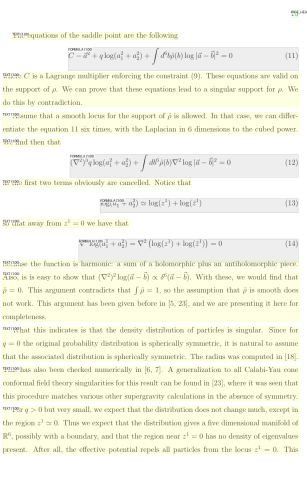
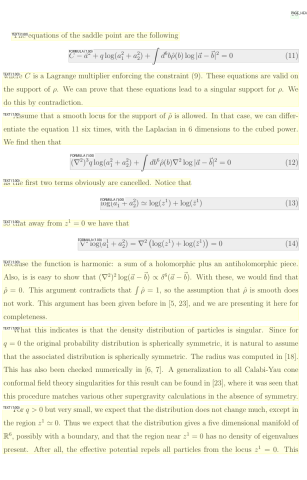
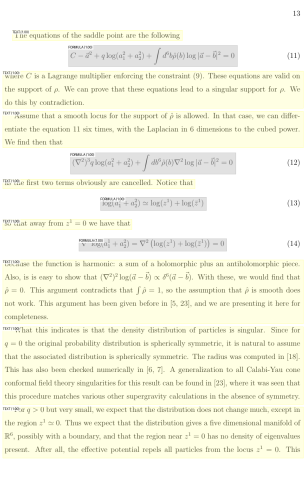
To illustrate and compare the visual grounding capabilities of SmolDocling, a set of sample predictions with SmolDocling and Qwen2.5-VL from DocLayNet [70] is shown in Table 7. Note that the element location results are independent from the correct reproduction of document content and structure.

Table 7: Visualizations of layout output from SmolDocling and QwenVL-2.5 compared to the DocLayNet ground truth. Examples are chosen to be representative of different layout styles and features. The prediction results however do not represent a generalizable measure of the model's performance on inputs with similar features. (1) Multi-column pages are handled by SmolDocling and Qwen2.5-VL, with some recall errors in the latter. (2) A manual page with terminal output shows poor bounding box recall on SmolDocling and label confusion in Qwen2.5-VL. (3) Lists with nesting are handled well in SmolDocling but confuse Qwen2.5-VL. (4) Both SmolDocling and Qwen2.5-VL reconstruct equations well, however with different annotation conventions on including or excluding the equation index. (5) On a portrait page with colorful elements and gradient background, SmolDocling creates less accurate bounding-boxes and Qwen2.5-VL suffers low recall. (6) On a report page with tables and diagrams, SmolDocling output exhibits some repetition loop, fabricating non-existent text cells (bottom left), while Qwen2.5-VL is confused between tables and pictures.

#	Ground-Truth	SmolDocling	Qwen2.5-VL	ID
1				0b49b1c3d5217411a7c9b92d960d5f61
2				0aafe8d4b66ac7ab81e9456327fbdb01

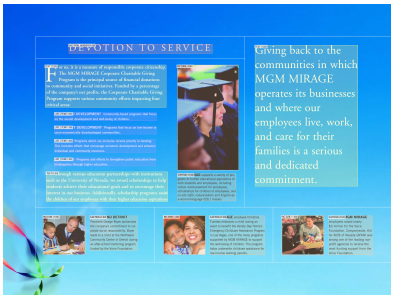
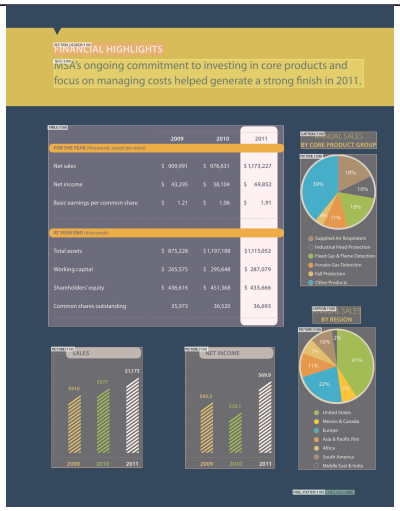
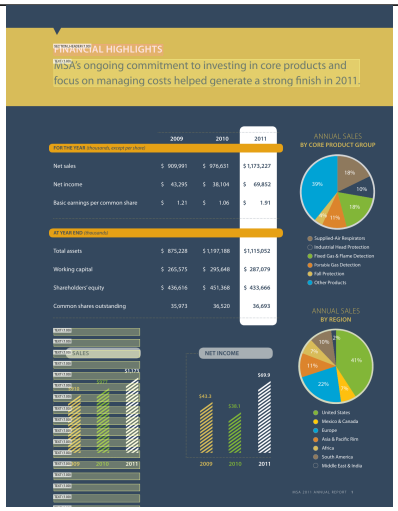
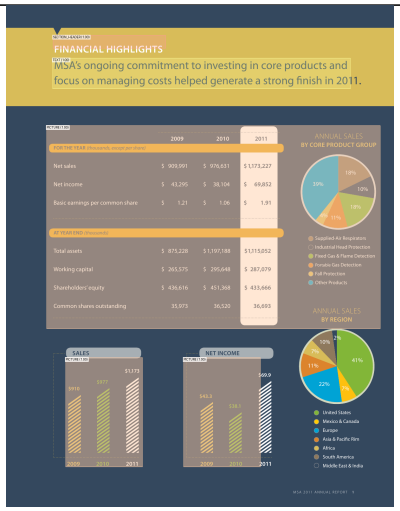
Continued on next page

Table 7: Visualizations of layout output from SmolDocling and QwenVL-2.5 compared to the DocLayNet ground truth. Examples are chosen to be representative of different layout styles and features. The prediction results however do not represent a generalizable measure of the model's performance on inputs with similar features. (1) Multi-column pages are handled by SmolDocling and Qwen2.5-VL, with some recall errors in the latter. (2) A manual page with terminal output shows poor bounding box recall on SmolDocling and label confusion in Qwen2.5-VL. (3) Lists with nesting are handled well in SmolDocling but confuse Qwen2.5-VL. (4) Both SmolDocling and Qwen2.5-VL reconstruct equations well, however with different annotation conventions on including or excluding the equation index. (5) On a portrait page with colorful elements and gradient background, SmolDocling creates less accurate bounding-boxes and Qwen2.5-VL suffers low recall. (6) On a report page with tables and diagrams, SmolDocling output exhibits some repetition loop, fabricating non-existent text cells (bottom left), while Qwen2.5-VL is confused between tables and pictures.

#	Ground-Truth	SmolDocling	Qwen2.5-VL	ID
3				1bff94fbe34e976d53eabc5b8b43f355
4				2ab06e940ef25ff7b2e832ab8176de15

Continued on next page

Table 7: Visualizations of layout output from SmolDocling and QwenVL-2.5 compared to the DocLayNet ground truth. Examples are chosen to be representative of different layout styles and features. The prediction results however do not represent a generalizable measure of the model's performance on inputs with similar features. (1) Multi-column pages are handled by SmolDocling and Qwen2.5-VL, with some recall errors in the latter. (2) A manual page with terminal output shows poor bounding box recall on SmolDocling and label confusion in Qwen2.5-VL. (3) Lists with nesting are handled well in SmolDocling but confuse Qwen2.5-VL. (4) Both SmolDocling and Qwen2.5-VL reconstruct equations well, however with different annotation conventions on including or excluding the equation index. (5) On a portrait page with colorful elements and gradient background, SmolDocling creates less accurate bounding-boxes and Qwen2.5-VL suffers low recall. (6) On a report page with tables and diagrams, SmolDocling output exhibits some repetition loop, fabricating non-existent text cells (bottom left), while Qwen2.5-VL is confused between tables and pictures.

#	Ground-Truth	SmolDocling	Qwen2.5-VL	ID
5				ebd1b976af12cd1d16d589390499 20df
6				eae40317d83ba98d3e70427e6d7baff6