LaTeX2Layout: High-Fidelity, Scalable Document Layout Annotation Pipeline for Layout Detection

Feijiang Han, Zelong Wang, Bowen Wang, Xinxin Liu, Skyler Cheung, Delip Rao, Chris Callison-Burch, Lyle Ungar

[1]University of Pennsylvania, Philadelphia, PA, USA
{feijhan, zew, bwwang, starliu, skcheung, delip, ccb, ungar}@seas.upenn.edu

Abstract

General-purpose Vision-Language Models (VLMs) are increasingly integral to modern AI systems for document understanding, yet their ability to perform fine-grained layout analysis remains severely underdeveloped. Overcoming this limitation requires large-scale, high-fidelity training datasets. However, current annotation methods that rely on parsing rendered PDFs are costly, error-prone, and difficult to scale. We propose a different paradigm: extracting ground-truth layout directly from the LaTeX compilation process rather than the final PDF. We present **`LaTeX2Layout`**, a generalizable procedural pipeline that recovers pixel-accurate bounding boxes and reading order from compiler traces. This enables the generation of a 140K-page dataset, including 120K programmatically generated synthetic variants that more than double the layout diversity of real-world data. Using this dataset, we fine-tune an efficient 3B-parameter VLM with an easy-to-hard curriculum that accelerates convergence. Our model achieves Kendall's $\tau = 0.95$ for reading order and mAP@50$= 0.91$ for element grounding, delivering nearly **200%** relative improvement over strong zero-shot baselines such as GPT-4o and Claude-3.7.

1 Introduction

The widespread adoption of Large Language Models (LLMs) and retrieval-augmented generation (RAG) systems has created urgent demand for accurate PDF document understanding (Gao et al. 2023; Sharma 2025). While PDF layouts, often encoding rich semantic information, are optimized for human readability, they pose significant challenges for automated processing (Xie et al. 2024). These challenges arise from parsing complex multi-column layouts, tables, figures, mathematical formulas, and cross-references (Zhong, Tang, and Jimeno Yepes 2019; Li et al. 2020; Dolfi et al. 2022; Sun et al. 2025) while preserving semantic relationships – an essential requirement for downstream applications (Tkaczyk, Zhong et al. 2022; ComPDF 2025; Liu et al. 2024; Wang et al. 2022). Document layout models like LayoutLM (Xu et al. 2020; Huang et al. 2022) and PP-DocLayout (Sun et al. 2025) depend on large-scale annotated datasets.

However, most existing datasets derive layout annotations directly from PDFs, facing two major challenges: **(1) Reliance on expensive annotations:** Manual annotation for document layout extraction (e.g., DocLayNet (Dolfi et al. 2022), D^{4}LA (Da et al. 2023)) is expensive at the scale necessary for training high-quality models. **(2) Error-prone semi-automatic methods:** Semi-automatic methods that combine PDF parsers with LaTeX/XML markup (e.g., PubLayNet (Zhong, Tang, and Jimeno Yepes 2019), DocBank (Li et al. 2020), and TableBank (Li et al. 2019b)) still depend on these parsers, which frequently mislocalize elements and produce incorrect reading orders, particularly in scientific documents (Adhikari and Agarwal 2025).[1]

To address these challenges head-on, we propose **`LaTeX2Layout`**, a procedural pipeline that produces pixel-perfect layout annotations for elements in a PDF document. We achieve this by exploiting a **key insight:** *The LaTeX compiler is aware of the bounding boxes and layout information during the compilation process, and this can be used to generate ground-truth annotations without additional parsing.*

Instead of treating layout extraction as a post-hoc vision problem over rendered pages, we instrument the LaTeX source to record structural metadata during compilation. This enables our method to obtain accurate layout and reading order annotations at scale. Moreover, LaTeX's modular source code enables procedural data generation for a diverse synthetic dataset creation. We can arbitrarily vary layouts, fonts, and element arrangements while programmatically obtaining aligned annotations for data augmentation.

Our primary contributions are:

1. **A cheap and reliable annotation technique**: We introduce **`LaTeX2Layout`**, a procedural layout annotation pipeline that extracts accurate layout annotations at 2.5 pages per second on commodity CPU hardware. Our pipeline can also be generalized beyond academic papers to enable annotations for other LaTeX-based documents, such as resumes, newspapers, and textbooks.
2. **A procedural data generation framework and principled training strategy for layout detection**: We de-

[1]See DocBank Issues 17 and 27:
https://github.com/doc-analysis/DocBank/issues/17
https://github.com/doc-analysis/DocBank/issues/27

sign programmatic LaTeX perturbations to create diverse document variants with matched annotations, yielding a synthetic dataset that offers roughly twice the layout diversity of real-world datasets. Furthermore, we demonstrate that applying curriculum learning by re-ranking our dataset significantly accelerates convergence and improves final model performance when fine-tuning general-purpose VLMs.

3. **State-of-the-art results for general-purpose VLMs**: We demonstrate the effectiveness of our data and methods by fine-tuning Qwen-2.5-VL-3B. Our resulting model achieves a Kendall's Tau of 0.95 for reading order and mAP@50 of 0.91 for element grounding. This represents a nearly **200% relative improvement** over zero-shot VLM baselines like GPT-4o and Claude-3.7. Our model also demonstrates superior robustness to visual perturbations and generalization to unseen elements compared to specialized models.

2 Related Work

Layout Dataset Construction via PDF Parsing

Existing layout annotation pipelines rely on external PDF parsers (PDFMiner, PDFPlumber, PyMuPDF) to extract layout metadata. PubLayNet (Zhong, Tang, and Jimeno Yepes 2019) aligns PubMed XML with PDFs, while DocBank (Li et al. 2020), DocGenome (Chen et al. 2024), and LaTeX Rainbow (Duan, Tan, and Bartsch 2023) use color-tagging in LaTeX source followed by PDF post-processing. These approaches enable scalability but inherit parser inaccuracies, particularly for complex layouts and mathematical content. Our method extracts layout metadata directly from the LaTeX compiler, eliminating parsing errors and providing deterministic annotations.

Synthetic Layout Data Generation

Prior synthetic generation methods employ deep generative models trained on real data. Early GAN-based approaches like LayoutGAN (Li et al. 2019a) and DocSynth (Biswas et al. 2021) produced low-resolution outputs with limited utility. Recent advances treat layout as sequence modeling: LayoutDM (Chai, Zhuang, and Yan 2023) uses diffusion on token sequences, while DocSynthv2 (Biswas et al. 2024) employs autoregressive transformers. Other approaches include GNNs for relational modeling (Agarwal et al. 2024), LLM-driven generation (Jiang et al. 2025), and algorithmic methods like 2D bin packing (Zhao et al. 2024). These methods require significant computational resources and struggle with complex layouts. By programmatically generating LaTeX source, we leverage a mature typesetting engine to produce structurally correct documents with perfect annotations as a deterministic compilation byproduct, eliminating expensive annotation steps.

Document Layout Analysis Models

Recent models jointly model textual, visual, and spatial features. The LayoutLM series (Xu et al. 2020, 2021; Huang et al. 2022) extends transformers with layout embeddings. LiLT (Wang, Jin, and Ding 2022) enables cross-lingual understanding, while LayoutLLM (Huang et al. 2023) integrates layout into LLMs. Detection-based approaches include DocLayout-YOLO (Zhao et al. 2024) and PP-DocLayout (Du et al. 2025). Despite their success, these models require domain-specific architectures. Although VLMs have been widely applied in various document understanding systems, the potential of general-purpose VLMs for layout understanding remains largely unexplored (Zhang et al. 2024; Zong et al. 2025). Our work addresses this gap by fine-tuning a general-purpose VLM with high-fidelity layout data, demonstrating significant improvements in grounding and reading order prediction.

3 The `LaTeX2Layout` Pipeline

While LaTeX compilation is deterministic, standard toolchains lack a native mechanism for exporting fine-grained layout metadata, such as element bounding boxes and reading order. To address this gap, we introduce **`LaTeX2Layout`**, a pipeline that instruments the compilation process itself to extract high-fidelity layout annotations. Our approach treats LaTeX as a programmable typesetting engine; we inject lightweight, layout-aware commands into the source code that instruct the compiler to emit precise spatial metadata during a single compilation pass.

The **`LaTeX2Layout`** pipeline operates in two stages:

- **LaTeX-Level Instrumentation:** Two custom packages – the Non-Text Element Tracker (NET) and the Text Line Tracker (TLT) – annotate source content to record layout metadata as part of the compilation process.
- **Post-Compilation Processing:** The emitted metadata is parsed to compute bounding boxes, align coordinate systems, and reconstruct reading order.

Tracking Non-Text Elements

The NET extracts layout metadata for non-paragraph elements in LaTeX documents. These include titles, headings, figures, tables, captions, and mathematical expressions. Such elements typically follow consistent syntactic patterns, making them well-suited for systematic instrumentation. Implementation details are provided in our Appendix.

Inspired by the LaTeX *\fbox* command that visually frames content, we introduce a novel command *\layoutmark* that silently records the layout of the wrapped element during compilation. It captures the following metadata: (a) Baseline Coordinates; (b) Box Dimensions, including width, height, and depth. [2]. All metadata is written to the auxiliary *.aux* file [3], enabling structured post-processing. Our Appendix provides the specific format of this compiler output.

Instrumentation Methods The NET is compatible with any context where the *\fbox* command can be applied. We

[2] In LaTeX, height refers to the portion above the baseline, and depth refers to the portion below.

[3] The `.aux` file is a compiler-generated side file typically used for cross-referencing. We extend it to store layout information.

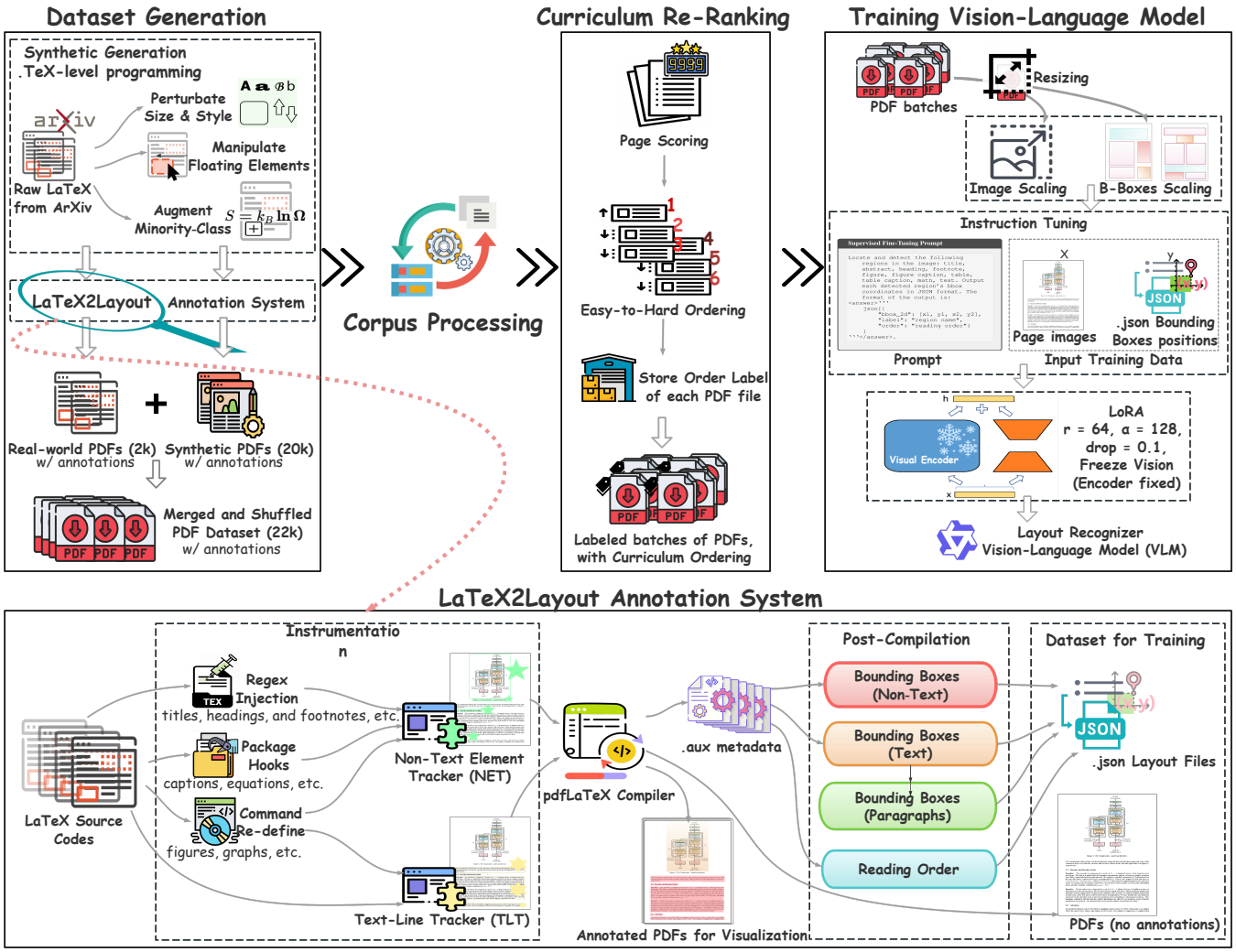

Figure 1: An overview of our pipeline for dataset generation and model training. The figure is split into the main workflow (top) and our core annotation system (bottom). **(1) Dataset Generation**: We crawl and synthetically perturb LaTeX source files from arXiv. Our **`LaTeX2Layout` Annotation System** (magnified below) compiles these sources, instrumenting them to extract pixel-perfect layout annotations directly from the compiler. **(2) Curriculum Re-Ranking**: The resulting dataset is scored by layout complexity and sorted into an easy-to-hard curriculum. **(3) VLM Training**: Finally, a VLM is instruction-tuned on the ordered curriculum to predict bounding boxes and reading order from page images.

support three flexible instrumentation strategies to accommodate the diverse range of elements. Full implementation examples are available in our Appendix.

- **Regex-Based Injection:** For elements with simple and consistent syntax, such as titles, abstracts, and headings, we use regular expressions to automatically find and wrap them with our *\layoutmark* command. However, this external matching approach can be brittle for elements with more complex or variable syntax. For these cases, we employ two more robust methods.
- **Package Hook Integration:** Many LaTeX packages provide internal hooks or options that we leverage for more reliable instrumentation. For example, the *caption* package allows us to define a custom format that automatically applies *\layoutmark* to every figure and table caption. Similarly, for complex multi-line mathematical content, the *empheq* package's *box* option can be hooked to guarantee a precise formula box.
- **Command Redefinition:** As a final and universally applicable strategy, we use LaTeX's native command redefinition mechanism. This approach is ideal for core commands like *\includegraphics* that are too syntactically diverse for regex and lack convenient package hooks. We redefine the target command to first wrap its content with *\layoutmark* and then execute its original functionality.

Tracking Text Lines

While NET works well for structured non-text elements, plain text paragraphs pose a different challenge. They lack explicit LaTeX markup, making them difficult to identify using pattern matching. To address this, we introduce the TLT. Inspired by the *lineno* package, which numbers lines in PDFs after compilation, TLT redefines its internal hook *MakeLineNo* to track line-level positional metadata automatically. This process requires no changes to the paragraph structure and no manual commands. Implementation details are provided in the Appendix.

Each text line is annotated with the following metadata: (a) Baseline Coordinates: the position of the line's lower-left corner; (b) Paragraph ID: a unique identifier for the LaTeX paragraph the line belongs to; (c) Column Number: the column index of the line in the rendered PDF, useful for detecting column breaks; (d) Line Height and Width: the vertical and horizontal extent of the rendered text. The exact format for this line-level metadata is also detailed in the Appendix.

Layout Extraction

The layout metadata emitted to the auxiliary (*.aux*) file is recorded in LaTeX coordinates. To recover accurate PDF-level layout, we convert units and transform axes. We also aggregate text lines into paragraph-level segments, with explicit handling of column and page breaks.

Bounding Box Computation For elements tracked by the NET, the bounding box is computed as:

$$\mathbf{B}_{\text{nontext}} = \begin{bmatrix} x_{\text{base}} \\ y_{\text{base}} + B_h \\ x_{\text{base}} + B_w \\ y_{\text{base}} - B_d \end{bmatrix} \tag{1}$$

Here, B_h is the height above the baseline, B_d is the depth below the baseline, and B_w is the element width.

For individual text lines tracked by TLT, the bounding box is computed as:

$$\mathbf{B}_{\text{text}} = \begin{bmatrix} x_{\text{line}} \\ y_{\text{line}} + L_h \\ x_{\text{line}} + L_w \\ y_{\text{line}} \end{bmatrix} \tag{2}$$

Here, L_h is the rendered line height, and L_w is the line width.

Paragraph-Level Aggregation Paragraphs are reconstructed by grouping text lines with the same paragraph ID. To capture higher-level semantics, adjacent paragraphs are merged if their vertical spacing falls below a configurable threshold. In practice, this threshold is defined as twice the height of a text line.

Cross-Column and Cross-Page Linking The TLT module tracks the column and page number of each line, enabling accurate reconstruction of paragraph structures that span across columns or pages. This allows the system to distinguish true paragraph breaks from layout-induced ones.

Coordinate Transformation LaTeX reports positions in scaled points (sp). We convert them to pixel coordinates at a target rendering DPI (dots per inch) via:

$$\begin{aligned} \text{Points (pt)} &= \frac{\text{Scaled Points (sp)}}{65536}, \\ \text{Inches} &= \frac{\text{Points (pt)}}{72.27}, \\ \text{Pixels} &= \text{Inches} \times \text{DPI} \end{aligned} \tag{3}$$

Here, DPI specifies the rendering resolution and defines the pixel-to-physical length ratio.

Additionally, the LaTeX coordinate system uses a bottom-left origin, whereas the PDF coordinate system uses a top-left origin. Therefore, the y-coordinate is adjusted as:

$$y_{\text{PDF}} = \text{Page Height} - y_{\text{LaTeX}} \tag{4}$$

Reading Order Alignment Unlike heuristic PDF parsers that assume fixed reading flows (e.g., top-to-bottom or left-to-right), we derive reading order directly from the compilation sequence. This approach preserves authorial intent and resolves ambiguous layout cases. For example, when a paragraph references a footnote, LaTeX compiles the referencing text line first, immediately followed by the footnote layout, even if the footnote appears at the bottom of the page.

4 Dataset Generation

Constructing the Real-World Seed Dataset

To build a seed corpus of contemporary scientific documents, we curated a 20K-page dataset by crawling LaTeX sources from arXiv papers published between 2022 and 2024. To capture a representative layout distribution, we balanced single-column formats (e.g., NeurIPS, ICML) and double-column formats (e.g., AAAI, ACL) from top venues. We then processed each compilable source with **`LaTeX2Layout`** to extract high-fidelity layout and reading-order annotations. The appendix provides a detailed breakdown of this dataset.

Synthetic Augmentation

Relying solely on crawled data poses two key challenges: the pipeline (crawling, cleaning, and validation) is labor-intensive, and the resulting corpus often has an imbalanced, long-tail distribution of element types that is hard to control. To address this, we introduce a procedural augmentation method that efficiently generates a more complex, diverse, and balanced dataset. Figure 2 shows example documents produced by this method, along with perfectly aligned annotations.

Our approach builds on the reusability of the trackers: once a LaTeX source is instrumented, we can generate new compiler-derived layout variants simply by modifying styles or reordering elements and recompiling. This avoids rerunning the full instrumentation pipeline for each variant and enables large-scale data generation. Using this procedure, we generated a 120K-page augmented dataset. As detailed in

the Appendix, this new corpus more than doubles the number of unique page-level element combinations and nearly doubles the aggregate proportion of critical minority classes. We employ three strategies:

- **Size and Style Perturbation (30K):** We apply controlled visual perturbations by scaling floating elements (figures, tables, equations) and adjusting page-level styles (e.g., margins, fonts). This enriches layout diversity while preserving the original semantic structure, creating novel layouts that are often more challenging for the model to parse than their original counterparts.
- **Floating Element Manipulation (80K):** Floating elements create rich layouts by displacing the natural flow of text. We amplify this effect by randomly reordering and duplicating floats within documents. These manipulations generate complex and varied float–text interactions, exposing the model to a wider range of challenging spatial arrangements.
- **Minority Element Augmentation (10K):** To directly counteract class imbalance, we generate pages composed exclusively of underrepresented elements (e.g., figures, tables, footnotes, formulas). This provides focused, targeted supervision for these critical minority categories, preventing them from being overlooked during training.

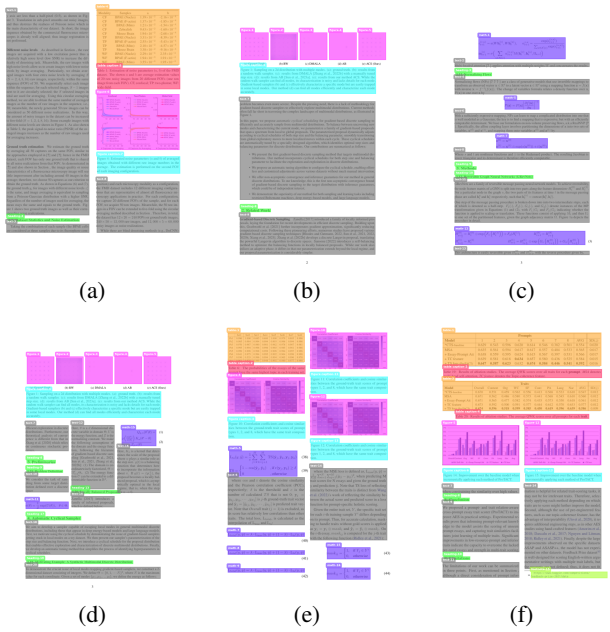

Figure 2: High-fidelity layout annotations produced by **`LaTeX2Layout`**. Top: examples from crawled real-world documents. Bottom: examples from procedurally generated synthetic documents. Additional synthetic samples and broader document types are provided in the Appendix.

Curriculum Learning Re-Ranking

Document layouts possess an intrinsic and measurable difficulty gradient, ranging from simple, single-column pages to dense layouts with complex, interwoven elements. We argue that this inherent difficulty gradient is ideally suited for leveraging the power of curriculum learning (CL)—a training paradigm that guides a model to learn progressively from simple to more complex examples to optimize final convergence and performance. While CL has proven effective in other domains (Bengio et al. 2009; Saha et al. 2024), our work extends this strategy to the task of document layout parsing.

Specifically, we re-rank our entire training dataset (both real and synthetic) by scoring page complexity based on the number and heterogeneity of its constituent elements. This graduated ordering, from simple to complex, allows the model to first master a foundational visual grammar—such as basic text blocks and headings—before progressing to more challenging structures like multi-column floats and dense tables. Empirically, we find that this approach not only accelerates convergence but also improves the final model's performance and training stability.

5 Model Training and Evaluation

Training. We adopt Qwen-2.5-VL-3B as the backbone of our vision-language system, given its strong performance and growing adoption in recent multimodal research (Bai et al. 2025). Our core training strategy involves freezing the pre-trained vision encoder and using Low-Rank Adaptation (LoRA) (hiyouga 2023) to efficiently fine-tune only the language model. This approach focuses the training on aligning the model's powerful visual features with our desired structured JSON output, which specifies each element's bounding box, semantic label, and reading order. The model was trained on our combined dataset of 140K document images. Further details on image preprocessing, prompt design, and specific hyperparameters are available in the Appendix.

Evaluation. Following previous works (Zhao et al. 2024; Chen et al. 2024), our evaluation assesses two distinct capabilities: element grounding and reading order prediction. We use Average Precision (AP) with a 0.5 IoU threshold to measure grounding accuracy and Kendall's Tau (τ) (Lapata 2006) [4] to evaluate the correctness of the predicted reading order. A key step in our pipeline is adapting the generative VLM for standard evaluation. Since the model does not produce explicit confidence scores, we propose a method to derive them from the token probability of the predicted class. These scores are then used with class-wise Non-Maximum Suppression (NMS) to refine detections before computing the final metrics. The full evaluation protocol is detailed in the Appendix.

6 Experimental Analysis

Evaluation Setup

Dataset. To ensure a realistic evaluation, we curated a test set of 1K real-world academic document pages following the

[4] Kendall's Tau (τ) is a rank correlation coefficient that measures the ordinal association between two sequences, ranging from -1 (perfect disagreement) to 1 (perfect agreement). This helps evaluate how well the predicted reading order matches the ground truth.

same data standard as the training set. Ground-truth annotations were derived using our **LaTeX2Layout** pipeline and were manually verified to ensure accuracy and consistency. Detailed statistics are available in the Appendix.

Models. We evaluate (1) zero-shot performance of general-purpos VLMs (GPT-4o (OpenAI 2024), Claude-3.7-Sonnet (Anthropic 2024), and Qwen-2.5VL-3B-Instruct (Qwen-2.5VL) (Bai et al. 2025)), (2) Qwen-2.5VL fine-tuned on 20K real and 140K (20K real + 120K synthetic) pages, and (3) YOLOv8 trained on the same real and synthetic datasets for grounding-only comparison.

Main Results

Our experiments validate the quality of annotations produced by **LaTeX2Layout** and quantify the impact of our synthetic data on downstream performance. We evaluate 11 element categories; full results are reported in Table 1.

We first find that general-purpose VLMs struggle with specialized layout parsing in a zero-shot setting. Strong models such as GPT-4o and Claude-3.7 achieve low reading-order τ (0.31 and 0.32) and modest element-grounding mAP50 (0.34 and 0.29), with particularly poor performance on fine-grained categories such as captions and footnotes. The smaller Qwen-2.5VL-3B performs even worse: due to limited pretraining for structured generation, it often fails to follow the prompt, producing incomplete or improperly formatted outputs. These results suggest that, without task-specific fine-tuning, general VLMs lack the spatial inductive biases needed for layout parsing.

Fine-tuning VLMs on layout-specific data derived from **LaTeX2Layout** substantially improves performance. Fine-tuning Qwen-2.5VL on our 20K real-world samples achieves a reading-order τ of 0.91 and an element-grounding mAP50 of 0.78. However, performance on rare or small categories (e.g., footnotes and math) remains limited, which we attribute to class imbalance and constrained layout diversity in the crawled corpus.

Adding our 120K synthetic samples mitigates this long-tail issue and further improves performance, reaching a final τ of **0.95** and mAP50 of **0.91**. The combined dataset yields large relative gains on previously challenging categories, including +53% for footnotes and +64% for math. Overall, these results highlight that the richer and more varied layouts introduced by our synthetic data are crucial for robust document understanding.

Ablation Studies

We conduct two ablation studies to isolate the contributions of our core components: synthetic data and curriculum learning.

Effectiveness of Synthetic Data: A Scaling Analysis To quantify the value of synthetic data, we train models on synthetic datasets of increasing size and compare them to a baseline trained on 20K real-world samples. Figure 3 highlights two key findings.

First, reading-order performance saturates quickly. Training on just 20K synthetic samples reaches Kendall's $\tau = 0.90$, nearly matching the real-data baseline ($\tau = 0.91$).

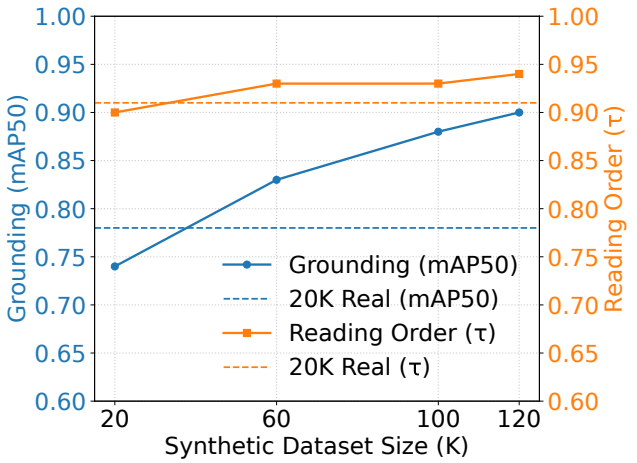

Figure 3: Impact of synthetic dataset size on grounding (mAP50) and reading order (τ) relative to a 20K real-data baseline.

This suggests that the structural logic of academic layouts is highly regular and can be learned from a moderately sized corpus.

Second, element grounding exhibits a clear scaling trend: mAP50 improves steadily as the amount of synthetic data increases. At 20K, the synthetic-only model slightly underperforms the real-data model, which we attribute to stronger **distributional alignment** between the real training set and the test distribution at small scale. As the synthetic corpus grows, however, the benefit of **structural diversity** becomes dominant, closing the gap and ultimately surpassing the real-data baseline.

Practical takeaway: *the gains are driven less by the visual novelty of individual elements than by the richness and complexity of the layouts they occupy. This validates programmatic data generation that composes existing elements into diverse structural arrangements.*

Effectiveness of Curriculum Learning To isolate the effect of curriculum learning, we train on the 20K real-world dataset using an easy-to-hard schedule and compare it to standard random-order training. The results show that CL improves both optimization and final accuracy.

First, CL improves training dynamics. As shown in the Appendix, curriculum training converges faster and reaches a lower final training loss, indicating a more stable learning process.

Second, this stability translates into better downstream performance (Table 2): reading-order τ increases from 0.88 to 0.91, and average grounding mAP50 improves from 0.72 to 0.78. The effect is most pronounced for challenging, low-frequency categories; for example, footnote mAP50 more than doubles from 0.28 to 0.57.

We hypothesize that CL induces a more efficient, compositional learning process. Under random-order training, the model must simultaneously learn element appearance and complex spatial relations on difficult pages, and losses from minority classes can be dominated by frequent ones. Our

Model	Reading Order (τ)	Element Grounding (mAP50)											
		Tit	Abs	Head	Txt	Math	Fig	Tab	FigCap	TabCap	Foot	Ref	Average
Zero-Shot Baselines													
GPT-4o	0.31	0.58	0.37	0.08	0.56	0.18	0.57	0.35	0.17	0.21	0.09	0.62	0.34
Claude-3.7	0.32	0.65	0.58	0.05	0.48	0.08	0.38	0.39	0.01	0.07	0.12	0.41	0.29
Qwen-2.5VL (Base)	0.12	0.09	0.00	0.07	0.21	0.17	0.09	0.03	0.03	0.00	0.06	0.00	0.07
Ours (Qwen-2.5VL Fine-tuned)													
w/ 20K Real Samples	0.91	0.75	0.85	0.73	0.92	0.55	0.82	0.85	0.84	0.78	0.57	0.92	0.78
+ 120K Synthetic Samples	**0.95**	**0.86**	**0.95**	**0.83**	**0.95**	**0.90**	**0.93**	**0.92**	**0.93**	**0.92**	**0.87**	**0.97**	**0.91**

Table 1: Main results for reading order prediction (τ) and element grounding (mAP50). Fine-tuning Qwen-2.5VL with data generated by **`LaTeX2Layout`** yields substantial gains over strong zero-shot general-purpose VLM baselines.

Training Strategy	Reading Order (τ)	Element Grounding (AP50)											
		Tit	Abs	Head	Txt	Math	Fig	Tab	FigCap	TabCap	Foot	Ref	Average
Standard Training (Random)	0.88	**0.77**	0.78	0.66	0.87	0.44	0.78	0.83	0.84	**0.81**	0.28	0.85	0.72
Curriculum Learning (Re-Ranking)	**0.91**	0.75	**0.85**	**0.73**	**0.92**	**0.55**	**0.82**	**0.85**	0.84	0.78	**0.57**	**0.92**	**0.78**

Table 2: Ablation study of curriculum learning on the 20K real-world dataset. CL improves both reading order prediction and overall element grounding accuracy. Best scores for each metric are shown in **bold**.

curriculum partially decouples these demands: the model first learns reliable detectors for common elements in simpler layouts, then reallocates capacity on harder pages to model intricate arrangements and rarer categories. Overall, these results show that CL is a simple yet effective strategy for improving layout model training.

7 Further Analysis and Discussion

To further characterize performance and practical utility, we conduct three additional analyses: (1) a comparison with specialized CV detectors, (2) an error analysis of the VLM, and (3) an efficiency analysis of both the model and the **`LaTeX2Layout`** pipeline. We summarize the main findings below; full details are provided in the Appendix.

1. **VLM Performance and Versatility.** Our VLM not only achieves grounding accuracy competitive with the specialized YOLOv8 detector but also offers significant advantages in versatility. It performs multi-task learning (simultaneously grounding elements and predicting reading order), demonstrates OOD generalization, and shows greater robustness to visual noise.
2. **Limitations and Future Directions.** Our error analysis identifies specific limitations that highlight clear paths for future improvement. These include the omission of very small objects (addressable via targeted data synthesis), inaccuracies for OOD layouts (fixable by expanding the training corpus using our **`LaTeX2Layout`**'s generalization capabilities), and sensitivity to severe visual noise (which can be improved with data augmentation). We also pinpoint a nuanced training challenge where the model's confidence doesn't always align with its spatial accuracy, pointing to the need for spatially-aware training objectives.
3. **Pipeline Efficiency.** From a practical standpoint, our data extraction pipeline is highly efficient, processing each page in just **0.435 seconds** on a standard consumer CPU. This confirms its suitability for creating large-scale datasets affordably and at scale.

8 Conclusion

We introduced **`LaTeX2Layout`**, a compiler-derived pipeline that converts instrumented LaTeX sources into pixel-accurate PDF layout annotations with ground-truth reading order. By leveraging compilation traces instead of heuristic PDF parsing, **`LaTeX2Layout`** produces perfectly aligned bounding boxes and reliably resolves ambiguous cases (e.g., floats, footnotes, and cross-column/page structures). Using this capability, we curated a 20K-page real-world seed corpus from contemporary arXiv papers and proposed a procedural augmentation framework that scales annotation via recomposition and recompilation, yielding an additional 120K-page synthetic corpus with richer layout diversity and better long-tail coverage.

Empirically, we showed that strong general-purpose VLMs are unreliable for layout parsing in zero-shot settings, while fine-tuning with **`LaTeX2Layout`** supervision yields substantial gains in both reading order and element grounding. Our ablations further show that reading-order learning saturates quickly, whereas grounding continues to improve with larger synthetic corpora, highlighting the importance of structural diversity. Looking forward, **`LaTeX2Layout`** enables two practical extensions: (1) automatically synthesizing LaTeX templates for non-scientific PDFs to broaden domain coverage, and (2) integrating OCR to support mixed pipelines that combine compiler-derived structure with text recognition when needed.

Acknowledgments

This research was developed with funding from the Defense Advanced Research Projects Agency's (DARPA) SciFy program (Agreement No. HR00112520300). The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Adhikari, N. S.; and Agarwal, S. 2025. A Comparative Study of PDF Parsing Tools Across Diverse Document Categories. arXiv:2410.09871.

Agarwal, A.; Patel, H.; Pattnayak, P.; Panda, S.; Kumar, B.; and Kumar, T. 2024. Enhancing document ai data generation through graph-based synthetic layouts. *arXiv preprint arXiv:2412.03590*.

Anthropic. 2024. The Claude 3 Model Family: Opus, Sonnet, Haiku. https://www.anthropic.com/research/claude-3-model-family.

Bai, S.; Chen, K.; Liu, X.; Wang, J.; Ge, W.; Song, S.; Dang, K.; Wang, P.; Wang, S.; Tang, J.; Zhong, H.; Zhu, Y.; Yang, M.; Li, Z.; Wan, J.; Wang, P.; Ding, W.; Fu, Z.; Xu, Y.; Ye, J.; Zhang, X.; Xie, T.; Cheng, Z.; Zhang, H.; Yang, Z.; Xu, H.; and Lin, J. 2025. Qwen2.5-VL Technical Report. *arXiv preprint arXiv:2502.13923*.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, 41–48.

Biswas, S.; Jain, R.; Morariu, V. I.; Gu, J.; Mathur, P.; Wigington, C.; Sun, T.; and Lladós, J. 2024. Docsynthv2: A practical autoregressive modeling for document generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8148–8153.

Biswas, S.; Riba, P.; Lladós, J.; and Pal, U. 2021. Docsynth: a layout guided approach for controllable document image synthesis. In *International Conference on Document Analysis and Recognition*, 555–568. Springer.

Chai, S.; Zhuang, L.; and Yan, F. 2023. Layoutdm: Transformer-based diffusion model for layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18349–18358.

Chen, Z.; Wang, S.; Wang, J.; et al. 2024. DocGenome: An Open Large-scale Scientific Document Benchmark for Training and Testing Multi-modal Large Language Models. *arXiv preprint arXiv:2406.11633*.

ComPDF. 2025. What is PDF Document Layout Analysis (DLA)? Available at https://www.compdf.com/blog/pdf-document-layout-analysis.

Da, C.; Luo, C.; Zheng, Q.; and Yao, C. 2023. Vision Grid Transformer for Document Layout Analysis. In *ICCV*.

Dolfi, M.; et al. 2022. DocLayNet: A Large Human-Annotated Dataset for Document-Layout Analysis. *arXiv preprint arXiv:2206.01062*.

Du, X.; Wang, B.; Zhao, Z.; et al. 2025. PP-DocLayout: A Unified Document Layout Detection Model with Global-to-Local Perception. *arXiv preprint arXiv:2503.17213*.

Duan, C.; Tan, Z.; and Bartsch, S. 2023. LaTeX Rainbow: Universal LaTeX to PDF Document Semantic & Layout Annotation Framework. In *Proceedings of the Second Workshop on Information Extraction from Scientific Publications*, 56–67. Bali, Indonesia: Association for Computational Linguistics.

Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; Wang, H.; and Wang, H. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).

hiyouga. 2023. LLaMA-Factory: Unified Efficient Fine-Tuning of 100+ LLMs. https://github.com/hiyouga/LLaMA-Factory.

Huang, S.; Xu, Y.; Cui, L.; et al. 2023. LayoutLLM: Layout-Aware Large Language Models for Document Understanding. *arXiv preprint arXiv:2309.00909*.

Huang, Y.; Xu, Y.; Li, L.; Cui, Y.; Zhang, J.; and Wei, F. 2022. LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 3727–3739.

Jiang, Z.-H.; Lin, C.-W.; Li, W.-H.; Liu, H.-T.; Yeh, Y.-R.; and Chen, C.-S. 2025. Relation-Rich Visual Document Generator for Visual Information Extraction. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 14449–14459.

Lapata, M. 2006. Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*, 32(4): 471–484.

Li, J.; Yang, J.; Hertzmann, A.; Zhang, J.; and Xu, T. 2019a. Layoutgan: Generating graphic layouts with wireframe discriminators. *arXiv preprint arXiv:1901.06767*.

Li, M.; Cui, L.; Huang, S.; Wei, F.; Zhou, M.; and Li, Z. 2019b. TableBank: A Benchmark Dataset for Table Detection and Recognition. *arXiv preprint arXiv:1903.01949*.

Li, X.; Xu, Y.; Cui, L.; Huang, S.; Wei, F.; Zhou, M.; Li, M.; and Wang, D. 2020. DocBank: A Benchmark Dataset for Document Layout Analysis. *arXiv preprint arXiv:2006.01038*.

Liu, Y.; et al. 2024. AutoIE: An Automated Framework for Information Extraction from Scientific Literature. *arXiv preprint arXiv:2401.16672*.

OpenAI. 2024. Hello GPT-4o. https://openai.com/index/hello-gpt-4o/. Accessed: 2025-05-26.

Saha, R.; Fahim, A.; Fyshe, A.; and Murphy, A. 2024. Exploring Curriculum Learning for Vision-Language Tasks: A Study on Small-Scale Multimodal Training. *arXiv preprint arXiv:2410.15509*.

Sharma, C. 2025. Retrieval-Augmented Generation: A Comprehensive Survey of Architectures, Enhancements, and Robustness Frontiers. *arXiv preprint arXiv:2506.00054*.

Sun, T.; et al. 2025. PP-DocLayout: A Unified Document Layout Detection Model to Accelerate Large-Scale Data Construction. *arXiv preprint arXiv:2503.17213*.

Tkaczyk, D.; Zhong, X.; et al. 2022. VILA: Improving Structured Content Extraction from Scientific PDFs via Visual Layout Grouping. *Transactions of the Association for Computational Linguistics*. Available at https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00466/110438/VILA-Improving-Structured-Content-Extraction-from.

Wang, J.; Jin, L.; and Ding, K. 2022. LiLT: A Simple yet Effective Language-Independent Layout Transformer for Structured Document Understanding. *arXiv preprint arXiv:2202.13669*.

Wang, Z.; et al. 2022. A Hybrid Approach for Document Layout Analysis in Document Images. *arXiv preprint arXiv:2404.17888*.

Xie, X.; Yan, H.; Yin, L.; Liu, Y.; Ding, J.; Liao, M.; Liu, Y.; Chen, W.; and Bai, X. 2024. WuKong: A Large Multimodal Model for Efficient Long PDF Reading with End-to-End Sparse Sampling. *arXiv preprint arXiv:2410.05970*.

Xu, Y.; Li, M.; Cui, L.; Huang, S.; Wei, F.; and Zhou, M. 2020. LayoutLM: Pre-training of Text and Layout for Document Image Understanding. *KDD*.

Xu, Y.; Xu, T.; Lv, Y.; Cui, L.; Lu, Y.; Wei, F.; and Zhou, M. 2021. LayoutLMv2: Multi-modal Pre-training for Visually-Rich Document Understanding. *arXiv preprint arXiv:2012.14740*.

Zhang, J.; Huang, J.; Jin, S.; and Lu, S. 2024. Visual Prompting in Multimodal Large Language Models: A Survey. *arXiv preprint arXiv:2409.15310*.

Zhao, Z.; Kang, H.; Wang, B.; and He, C. 2024. DocLayout-YOLO: Enhancing Document Layout Analysis through Diverse Synthetic Data and Global-to-Local Adaptive Perception. *arXiv preprint arXiv:2410.12628*.

Zhong, X.; Tang, J.; and Jimeno Yepes, A. 2019. PubLayNet: Largest Dataset Ever for Document Layout Analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 1015–1022. IEEE.

Zong, Y.; Zhang, Q.; An, D.; Li, Z.; Xu, X.; Xu, L.; Tu, Z.; Xing, Y.; and Dabeer, O. 2025. Ground-V: Teaching VLMs to Ground Complex Instructions in Pixels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.