Document Layout Analysis – DocVision

Rutvik Patel
Master of Science - Data Science
University of Maryland, College Park
College Park, USA
rvpatel@umd.edu

Prahar Modi
Master of Science - Data Science
University of Maryland, College Park
College Park, USA
pmodi08@umd.edu

Abstract

Document layout analysis is a critical step in converting scanned documents into structured digital content. This project, DocVision, aims to segment a scanned document page into meaningful regions (such as text paragraphs, titles, tables, figures, headers, and footers) and classify each region. We developed a hybrid pipeline that combines classical layout segmentation algorithms with machine learning classification. First, two well-known rule-based methods – the recursive X-Y cut and the DocStrum algorithm – were implemented to partition page images into candidate zones. We then extracted a set of descriptive features (geometric, spatial, and texture-based) from each zone and trained a LightGBM classifier to label each region into one of 8 predefined categories from the DocLayNet dataset. Additionally, we experimented with a deep learning approach using a Faster R-CNN object detector as an end-to-end alternative for layout detection. The proposed hybrid approach achieved a classification accuracy of 87.5% on the test set, with high precision for text blocks (F1 0.91) and page footers (F1 0.96). These results demonstrate the effectiveness of combining classic segmentation techniques with modern classification models for document layout analysis. We discuss the comparative performance of the classical pipeline versus the Faster R-CNN model and outline directions for future improvements.

Introduction

The physical layout of a document page conveys important structural information that is crucial for tasks like optical character recognition (OCR), content extraction, and digital archiving. Document layout analysis is the process of identifying and categorizing regions in a scanned document image, such as paragraphs of body text, section headings, images, tables, and other elements. Accurately segmenting these components is challenging due to diverse page layouts, multi-column formats, and the presence of complex elements (figures, formulas, etc.). Traditional methods for layout analysis include rule-based segmentation algorithms like the recursive X-Y cut , which recursively splits a page using whitespace projections, and the DocStrum algorithm , which clusters connected components into text lines and blocks. More recently, data-driven approaches employing deep learning object detection (e.g., Faster R-CNN) have achieved state-of-the-art results on large datasets (e.g., PubLayNet, DocBank, DocLayNet), though at the cost of requiring extensive training data and computational resources.

Objectives: The goal of this project (DocVision) is to develop a system that can automatically segment a scanned document page into its constituent regions and classify each region into a predefined category. The targeted categories in this work include Title, Section-Header, Text (body text/paragraph), Picture (figure), Table, Formula, Page-Header, and Page-Footer. By correctly labeling these zones, the system can produce a structured representation of the page (e.g., as a JSON or XML layout), which could be used in downstream processes like structured PDF generation or content reflow. A key objective is to compare classical layout segmentation techniques with modern deep learning-based detection, and to explore whether a hybrid approach can leverage the strengths of both.

Scope and Challenges: Our approach focuses on analyzing the layout structure of document images, without performing actual text recognition. We assume input documents are scanned pages (grayscale or binary images) and limit the analysis to the eight region types listed above. Other possible document elements (such as captions, footnotes, or callout boxes) are considered out-of-scope in our implementation. In early development, we also aimed to infer the reading order of the de-

tected zones; however, this feature was dropped in the final project phase to concentrate on improving segmentation and classification accuracy. Instead, the project scope was expanded to classify more granular categories (eight categories based on the DocLayNet dataset, up from five basic classes in the initial prototype). This decision introduced additional complexity since distinguishing between similar classes (e.g., a Title vs. a Section-Header, or separating Text paragraphs from list items) requires capturing subtle layout cues. Overall, the major challenges addressed include handling varied layout structures (from simple single-column pages to complex multi-region layouts), integrating multiple segmentation algorithms, and achieving high classification performance across all zone types.

Literature Review

Early research in document layout analysis employed heuristic algorithms tailored for printed documents. For instance, Berg demonstrated that high-precision text extraction from PDFs is possible by carefully leveraging layout structure, underscoring the need for reliable segmentation techniques. Classic algorithms such as the X-Y cut apply a top-down recursive segmentation, splitting pages via whitespace projections to isolate text columns and blocks. Another foundational technique is the DocStrum approach introduced by O'Gorman, which uses connected-component clustering to group characters into text lines and then aggregates lines into text blocks. Variants of this idea have been explored by researchers such as Yu and Cho , who proposed a 3D neighborhood graph model for word extraction in machine-printed documents. These traditional methods do not require training data and can be very fast, but they rely on assumptions (e.g., clear separations between columns or consistent text line structure) that do not always hold for complex or diverse layouts.

In recent years, the advent of large annotated datasets and deep learning has significantly advanced layout analysis. Public datasets like PubLayNet and DocLayNet have enabled training convolutional neural networks to directly detect layout regions as object detection tasks. For example, the Faster R-CNN model – originally developed for natural object detection – has been applied to document images to localize zones such as figures, tables, and text blocks with high accuracy. These data-driven approaches often outperform rule-based algorithms on benchmark metrics, especially for challenging layouts. However, pure deep learning solutions require substantial computational resources and may generalize poorly to document styles not seen in training data.

Identification of Gaps: There is a middle ground in the literature that has not been fully explored: combining the precision of classical segmentation algorithms with the adaptability of learning-based classification. This project aims to fill that gap by using classical methods to segment a page and a machine learning model to classify each segment, thereby leveraging prior knowledge of layout geometry while still learning data-driven class distinctions. We also compare this hybrid pipeline against an end-to-end Faster R-CNN approach to highlight the trade-offs between interpretable, modular pipelines and monolithic deep networks.

Methodology

Data Collection: We utilized the DocLayNet core dataset—a rigorously curated, fully human-annotated corpus of 80,863 PDF pages—as the sole source of training and evaluation data. The pages come from six real-world document genres (financial reports, scientific articles, patents, government tenders, legal regulations, and user manuals) and were rasterised to uniform 1025×1025 px PNG images. Professional annotators drew every bounding box by hand, and a subset of pages received double- or triple-annotation to quantify label consistency; the resulting regions are stored in COCO-formatted JSON with 11 canonical layout classes. From these 11 classes we retained eight core categories—Title, Section-Header, Text, Picture, Table, Formula, Page-Header and Page-Footer—discarding Caption, Footnote and List-item to keep the label set focused and sufficiently populated. We adhered to DocLayNet's official, non-overlapping train / validation / test splits (69,375 / 6,489 / 4,999 pages respectively) to avoid layout-style leakage. Ground-truth boxes and their category_id labels served two purposes: (i) supervising the LightGBM classifier and (ii) providing an unbiased benchmark for our test-set evaluation.

Preprocessing: Each page image is first converted to grayscale and then binarized using Otsu's thresholding (with inversion, so text appears as white foreground on a black background). This step removes color information and normalizes the input, making subsequent segmentation more robust to varying illumination. Minor noise removal

and skew correction are implicitly handled by the thresholding process.

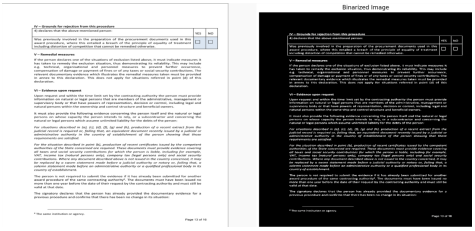

Figure 1: Preprocessing

Segmentation Algorithms: We implemented three layout segmentation approaches:

- Recursive X-Y Cut: This algorithm computes horizontal and vertical projection profiles of the binary page image (sums of pixel values along each row and column). It identifies large whitespace gaps and splits the page at the largest gap, either vertically or horizontally. The procedure recurses on each resulting sub-region until no further splits are found or a minimum block size is reached. The output is a set of rectangular regions (bounding boxes) corresponding to blocks of content.

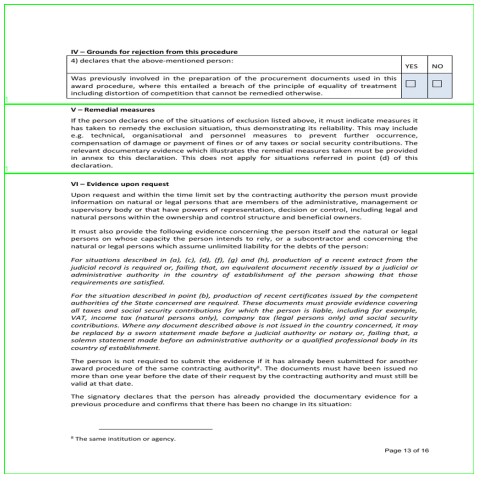

Figure 2: X-Y Cut

- DocStrum: This bottom-up approach analyzes connected components (e.g., individual characters) in the binary image. A k-nearest neighbor graph is constructed among components to estimate the typical orientation of text lines and the average character spacing. It then clusters the components into text lines and further groups lines into text blocks using a dynamic morphological merging step. The result is a set of bounding boxes around textual regions. DocStrum is effective at finding dense text regions but may produce many small boxes in non-text areas.

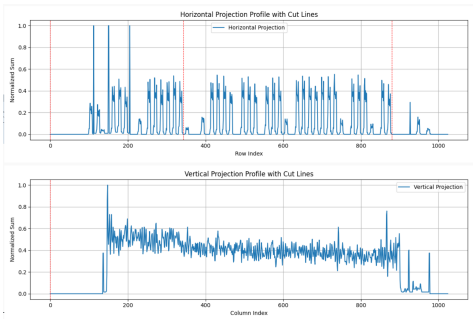

Figure 3: X-Y Cut

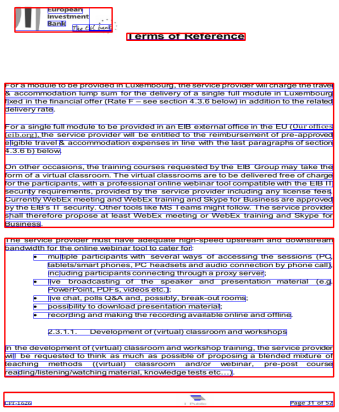

Figure 4: DocStrum

- Hybrid: We devised a hybrid method to leverage the strengths of both X-Y cut and DocStrum. First, X-Y cut is applied to partition the page into large blocks. Next, each block is classified as text-heavy or non-text by counting the number of small connected components within it (a high count implies a text-heavy block). For text-heavy blocks, we apply the DocStrum procedure inside that block to obtain fine-grained text lines, then merge those lines into paragraph-level boxes. Non-text blocks (e.g., large pictures or tables) remain as single regions from the X-Y cut. Finally, we combine all regions into one set of bounding boxes. This hybrid approach preserves large distinct regions while refining the segmentation within text-dense areas.

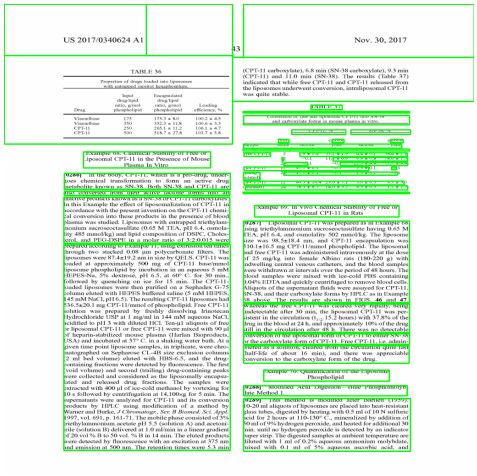

Figure 5: Hybrid Approcah

Feature Extraction: For each candidate region output by the segmentation step, we compute a set of features to characterize its geometry and content. The features include:

- Geometric – the region's width, height, aspect ratio (width/height), and area. These capture the shape and size; for example, very wide and short regions might indicate headers or footers, while tall narrow regions might indicate sidebars or separators.
- Spatial – the normalized coordinates of the region's center (x-center and y-center relative to the page dimensions). This encodes position on the page, since certain elements (e.g., titles or footers) appear in predictable locations (top or bottom of the page).
- Intensity – grayscale intensity statistics such as the mean, median, and standard deviation of pixel values in the region. This helps differentiate text (which tends to be high-contrast black on white) from images or graphics (which have more varied intensity distributions).
- Edge Density – the density of edges in the region, computed as the fraction of pixels detected as edges (using a Canny edge detector). Text regions typically yield many edges from character strokes and boundaries, whereas image or whitespace regions have fewer edges.
- Text Density – the proportion of dark (ink) pixels in the region after binarization. A high text density indicates a lot of printed text covering the area, distinguishing text blocks from areas of whitespace or light content.

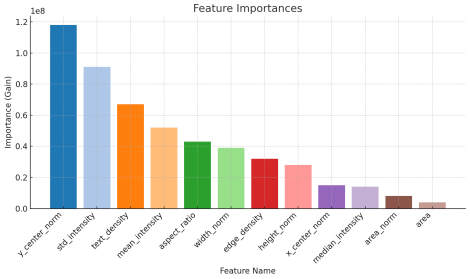

Figure 6: Feature Importance

All feature values were normalized (standardized) across the dataset. While we initially extracted about 11 features, we reduced this to the 8 most informative features for the final model to avoid redundancy. Here in the correlation matrix of all the features.

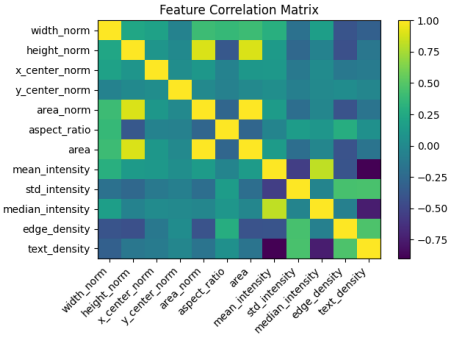

Figure 7: Correlation Matrix

Classification Model: We treat region classification as an 8-class problem. Our classifier of choice is LightGBM, a gradient-boosted decision tree model, which we selected for its speed and ability to handle numeric feature inputs effectively. We trained the LightGBM model using the labeled regions from the training set. We performed 3-fold cross-validation on the training data, using early stopping (if validation loss did not improve for 50 rounds) to determine the optimal number of boosting iterations. The final model was then trained on the full training+validation data with that number of iterations. During inference, the classifier outputs a probability distribution over the 8 classes for each region, and we take the highest-probability class as the prediction. In cases where the segmentation produced overlapping or nested candidate boxes, we applied simple heuristics to remove duplicates (preferring the larger box or the higher-confidence prediction) before finalizing the output.

Deep Learning Baseline: As a baseline for comparison, we also implemented an end-to-end layout analysis approach using a deep neural network. We fine-tuned a Faster R-CNN (with a ResNet-50-FPN backbone) on our dataset to detect the same 8 classes of regions. The model was initialized with pre-trained weights and trained for a limited number of epochs given computational constraints. While the Faster R-CNN can in principle learn to directly predict bounding boxes and classes for all regions in one step, our implementation was relatively small-scale. This experiment allows us to compare a fully learned method with our hybrid pipeline approach.

Evaluation Strategy: We evaluated the hybrid pipeline by comparing its outputs on test pages to the ground-truth annotations. A predicted region was considered correct if it overlapped sufficiently with a ground-truth region of the same class (using an Intersection-over-Union threshold to match predictions to truth). Using these matches, we computed standard classification metrics: overall accuracy (the percentage of regions correctly classified) and the precision, recall, and F1-score for each class. Our primary focus was on the hybrid approach's classification accuracy and per-class F1-scores, which we report in the next section.

Results

We tested our full pipeline on the DocLayNet test set. The hybrid segmentation+classification model achieved an overall classification accuracy of 87.47% on the test pages. Table 1 presents the F1-scores for each of the 8 layout categories.

Class	F1 Score
Text	0.91
Page-footer	0.96
Page-header	0.86
Picture	0.82
Section-header	0.8
Formula	0.84
Table	0.62
Title	0.68

Figure 8: Table 1

The classifier performed best on Text and Page-Footer regions, with F1-scores of 0.91 and 0.96 respectively, indicating very high precision and recall for those classes. Page-Header and Formula zones also achieved strong F1 scores around 0.85–0.86. On the lower end, Table zones proved the most challenging (F1 0.62), and Title regions also had a relatively lower F1 (0.68). The lower accuracy for tables is likely due to their complex structure (grids and multi-line content) which was sometimes segmented or classified imperfectly, while titles were occasionally confused with section headers or normal text. Overall, the model is highly effective for the frequent and distinctive layout elements, and acceptable but less accurate for the more complex or infrequent categories. In addition to these quantitative results, we examined the segmentation outputs qualitatively. We found that the pure X-Y cut segmentation often left some content under-segmented (for instance, it might group a figure and its caption together as one block or fail to separate adjacent text columns when the gap was small). The DocStrum segmentation, in contrast, produced many small regions (each line of text as a separate box) and sometimes over-segmented areas that should be a single region. The hybrid approach offered a balance: it kept obvious non-text elements (figures, tables, etc.) as distinct blocks while simultaneously segmenting dense text areas into appropriate paragraphs. Figure 1 shows an example output from our system on a sample page, with each detected region outlined and color-coded by its class label. Visually, the hybrid method yields a coherent page decomposition that aligns well with human expectations of the document's structure. (If a visual were included, Figure 1 would illustrate a page image with color-coded bounding boxes for regions like Title, Text, Table, etc.) The Faster R-CNN model that we trained for comparison did not perform as well as the hybrid pipeline. It tended to miss small regions like page headers or misclassify some regions (for example, sometimes a table was detected but labeled as a picture). Its overall precision and recall were substantially lower; for instance, many predictions had localization or labeling errors that led to a much lower mAP. This outcome is not surprising given the limited training we could perform – a fully-trained deep model on a large dataset might have surpassed our hybrid approach, but in our experiments the classical segmentation plus LightGBM classifier proved more effective and reliable.

Discussion

The high F1-scores for common text regions and footers indicate that our feature set and classifier were well-suited to those classes. These elements are relatively uniform across documents (for example, body text paragraphs have similar texture and

edge patterns, and footers are consistently located at the page bottom with small text), so the model was able to capture their characteristics strongly. In contrast, the poor result for tables highlights a limitation: tables have a complex visual structure that sometimes fooled the segmentation (occasionally splitting a single table into multiple boxes or merging it with nearby text) and thus led to classification errors. Similarly, distinguishing titles from section headers proved difficult because both often appear at the top of a page in large font; without textual content analysis or more context, the classifier had to rely on subtle size/position differences, which was not always reliable.

A key outcome of this project is the confirmation that a hybrid approach combining rule-based algorithms with machine learning can be highly effective for layout analysis. The classical segmentation algorithms provided a strong initial partitioning of the page based on explicit rules, which reduced the search space for the classifier. The LightGBM model then added adaptability, compensating for variations where fixed rules were insufficient. This two-stage approach gave us interpretability (we can distinguish segmentation errors from classification errors) and robustness on our dataset. On the other hand, the experiment with the Faster R-CNN detector underscores the dependency of deep models on large training data and careful tuning. Our Faster R-CNN was under-trained and thus struggled, whereas the hybrid system worked well with the available data. This does not mean deep learning is unsuitable for the task—rather, it suggests that in data-limited scenarios, a hybrid pipeline can outperform a quickly-trained end-to-end model.

Broader Implications: The success of our approach suggests that classic computer vision techniques, when thoughtfully combined with modern machine learning, still have considerable value. In industrial document processing applications, it may not always be feasible to train a large neural network for every new type of document layout. Our method demonstrates an alternative that achieves high accuracy with modest data and computational requirements. Furthermore, the modular design (segmentation + classification) means the system can be updated or extended more easily (for example, plugging in a new classifier or adding a rule for a new layout element) compared to a monolithic deep network. This is beneficial for real-world deployments where transparency and maintainability are important.

Limitations and Future Work: The current implementation is limited by the predefined set of layout zones and the quality of segmentation. If a document contains an element outside our 8 classes (for example, a sidebar or a chart), the system will not recognize it. Making the model more flexible or training it on additional classes (given sufficient data) would increase its applicability. Segmentation errors were a source of some classification mistakes; thus, one avenue for improvement is to incorporate more advanced segmentation methods. For instance, using a trainable segmentation network like Mask R-CNN could directly produce precise regions for each class. Another potential improvement is data augmentation or synthetic data generation for under-represented classes (especially tables and titles) to help the classifier learn those patterns better. We also dropped the reading order inference in this project due to time constraints; reintroducing a module to sequence the detected regions could enhance the output for use cases like PDF reconstruction or e-reading. Addressing these limitations would make DocVision more robust and comprehensive in analyzing document layouts.

Conclusion

In this project, we designed and implemented a system for automatic document layout analysis that successfully segments and classifies regions in scanned pages. The proposed DocVision pipeline, which integrates recursive X-Y cut segmentation, DocStrum clustering, and a LightGBM classification model, achieved an overall accuracy of approximately 87% on the DocLayNet benchmark with robust performance on key region types. We also explored a deep learning alternative using Faster R-CNN, finding that our hybrid approach outperformed it under constrained training conditions. The project's contributions include a demonstration of how classical and modern methods can be hybridized to tackle a structured vision problem, as well as an empirical evaluation of this approach against a neural network baseline. These findings reinforce the notion that incorporating domain-specific algorithms into an AI pipeline can yield efficient and interpretable solutions without requiring extremely large datasets. In summary, Document Layout Analysis – DocVision provides a valuable step toward fully automated understanding of document images, and its insights can guide future developments in both research and practical applications of document image processing.

References

1. Ø. R. Berg (2011). *High precision text extraction from PDF documents* (M.S. thesis, University of Oslo).
   Retrieved from https://www.duo.uio.no/handle/10852/24445

2. Y.-J. Yu & H.-G. Cho (2003). A word extraction algorithm for machine-printed documents using a 3D neighborhood graph model. *International Journal on Document Analysis and Recognition, 4*(2), 115–130.
   doi: 10.1007/s100320300006

3. UglyToad/PdfPig Project. *Document-Layout-Analysis Wiki* (description of X-Y cut algorithm). GitHub.
   https://github.com/UglyToad/PdfPig/wiki/Document-Layout-Analysis

4. B. Pfitzmann, C. Auer, M. Dolfi, A. S. Nassar & P. W. J. Staar (2022). *DocLayNet: A Large Human-Annotated Dataset for Document-Layout Analysis*. In Proc. ACM SIGKDD.
   doi: 10.1145/3534678.3539484

5. S. Ren, K. He, R. Girshick & J. Sun (2015). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. In Proc. NeurIPS.
   https://arxiv.org/abs/1506.01497