InvoiceAI:Extracting structured tables from documents using deep learning and OCR clustering

Jaanhavi lalingkar

.Department of Computer Engineering and Technology
MIT- World Peace University
Pune, India
jaanhavilalingkar@gmail.com

Abstract— Table extraction from scanned documents like invoices and receipts is crucial for automated data processing. Traditional OCR methods often struggle with complex layouts and noisy inputs. This paper introduces a deep learning-based end-to-end system that combines semantic segmentation with classical image processing and OCR to extract structured tables. The model integrates GoogLeNet-based feature extraction with custom decoders and a post-processing pipeline that clusters column positions and extracts textual data. By integrating modern architectures and classical CV techniques, the proposed system improves precision and reduces error propagation, making it suitable for business, legal, and healthcare applications[1][2].

Link to github

https://github.com/jaanhay/GNet-Table

Keywords:

- ***Table Extraction***
- ***Deep Learning***
- ***GoogLeNet***
- ***OCR***
- ***TableScope,***
- ***Tesseract***
- ***Semantic Segmentation***
- ***Column Clustering***
- ***Document Digitization***

INTRODUCTION

With the growing reliance on digital workflows across industries and government sectors, the demand for automated data extraction from scanned documents has become increasingly vital. Among these, extracting tabular information from invoices, receipts, and forms presents a particularly challenging problem due to inconsistent layouts, noisy scans, and varying resolutions.

Traditional rule-based or template-driven methods often fail to generalize across different document types. To address these limitations, this paper introduces **TableScope**, a comprehensive deep

learning-based framework for precise table detection and extraction. It leverages GoogLeNet for robust feature extraction, classical computer vision techniques for structural segmentation, and OCR for content recognition. Designed for adaptability, the system effectively handles diverse and complex document formats, offering a scalable and high-accuracy solution for real-world applications.

Problem statement

In today's data-driven landscape, organizations frequently rely on scanned documents such as invoices, receipts, and forms to store critical tabular information. Manual extraction of this data is not only time-consuming but also prone to human error, particularly when dealing with large volumes of documents.

Existing solutions often fall short in their ability to generalize across diverse document formats, especially when those documents contain noisy backgrounds, variable lighting, or non-standard table structures. Moreover, traditional OCR and rule-based systems are limited in handling complex or inconsistent layouts, leading to incomplete or inaccurate data extraction.

System architecture

1.Table Detection Using GoogLeNet-Based Encoder-Decoder Architecture:

The detection module utilizes an encoder-decoder framework based on GoogLeNet (InceptionV1), renowned for its computational efficiency and strong feature representation. The encoder comprises a pre-trained GoogLeNet model with its final classification layers removed, transforming it into a robust extractor of spatial and semantic features from input document images.

The decoder branches into two parallel pathways: one specialized in identifying the boundaries of table regions, and the other focused on detecting column structures within those regions. This bifurcated approach allows the network to learn nuanced representations for both tasks independently, resulting in more accurate and coherent segmentation of complex tabular layouts.

Hyperparameter Settings

I. **Optimizer: Adam**

II. **Initial Learning Rate: 0.0001**

III. **Learning Rate Schedule: Exponential Decay**
lr = 0.0001 × (0.96)^(epoch / 10)

IV. **β_1 (Adam): 0.9**

V. **β_2 (Adam): 0.999**

VI. **Loss Function: Binary Crossentropy *(assumed)***

VII. **Input Image Size: 128 × 128 × 3**

VIII. **Output Heads:**

IX. **`table_output`, `column_output`**

X. **Batch Size: 2**

XI. **Epochs: 50**

2. Classical Image Processing for Column Detection:

- Grayscale conversion and Otsu's binary thresholding
- Morphological dilation using vertical kernels
- Contour extraction for column band detection
- x-coordinates of contours are clustered using a gap threshold
- Final vertical column lines derived from each cluster

3. OCR and Output Extraction:

- Contours grouped vertically to identify rows
- Column alignment using x-centroids matched against detected lines
- Morphological cleaning of each cell image before OCR
- Tesseract OCR used for final text extraction

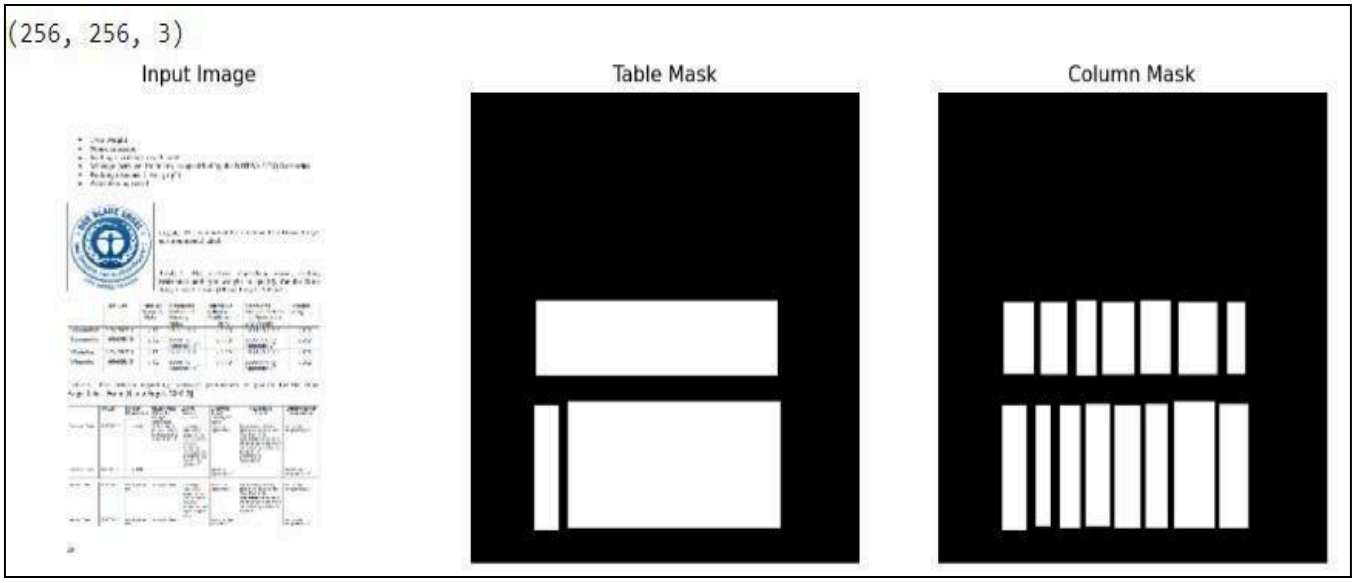


Fig : Marmot dataset annotations

Marmot Dataset

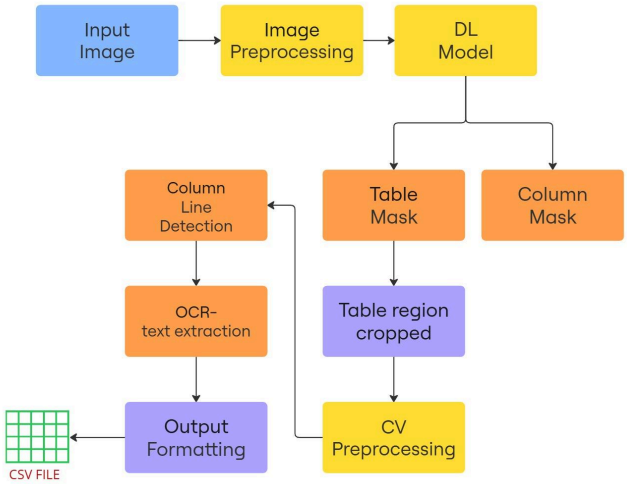
The primary dataset used in this research is a modified version of the Marmot Dataset, which has been tailored specifically for table detection (TD) and table structure recognition (TSR)[2][3] tasks. Originally comprising English document pages sourced from the CiteSeer repository, this dataset has been further enriched with detailed annotations. These include precise bounding boxes for entire table regions as well as additional column-level mask annotations, making it well-suited for supervised training of segmentation-based models.

The annotations facilitate both high-level detection of tabular zones and granular recognition of column divisions within each table. By providing diverse examples of document layouts and structures, the dataset supports the development of models that generalize well across real-world documents. This dataset is publicly accessible for academic and research purposes at: https://www.icst.pku.edu.cn/szwdclyjs/sjzy/index.htm.

Proposed Solution

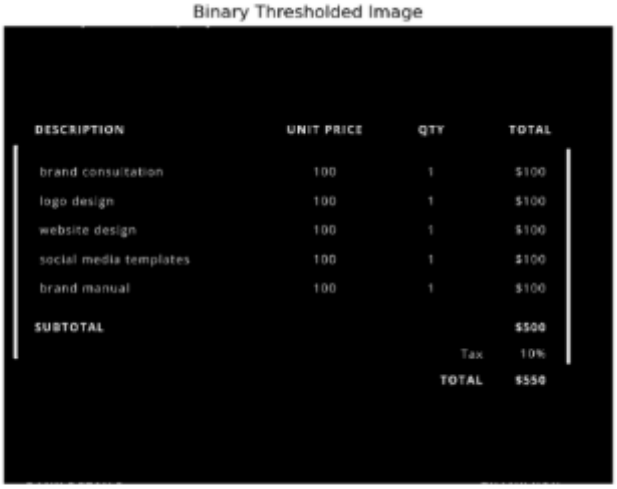
TableScope system combines GoogLeNet-based deep learning with classical CV techniques. The segmentation outputs from the decoder are post-processed using dilation, clustering, and contour analysis to form table boundaries and column lines. Text is extracted using OCR after alignment and cleaning.

1. Input Image
2. → Preprocessing for DL input
3. → Table Mask (Deep Learning)
4. → Preprocessing for CV
5. → Column Line Detection (Morphology + Clustering)
6. → OCR
7. → Output Formatting

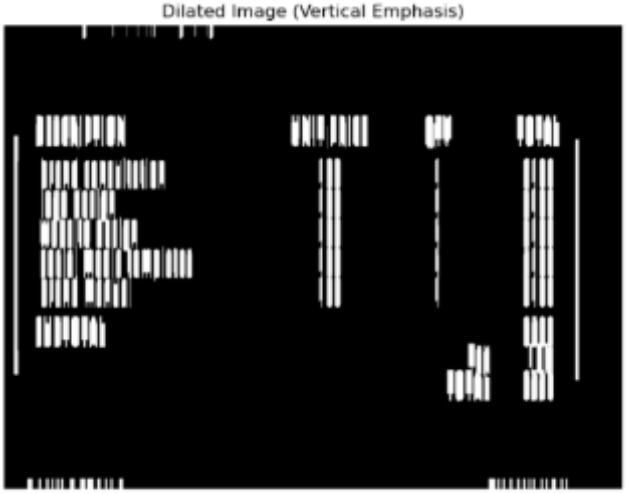


Predicted masks according to the stages:

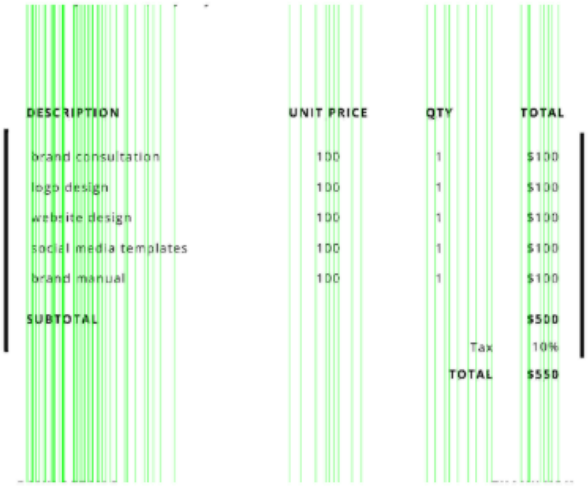
Binary Thresholded Image



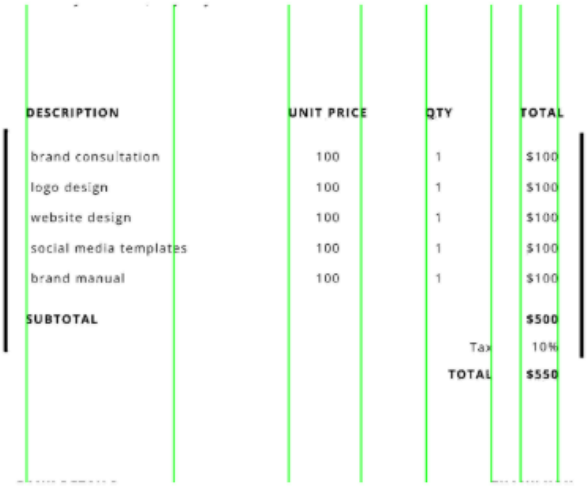
Dilated Image (Vertical Emphasis)



All Detected Green Lines



Final Clean Column Lines



3.5 OCR + Output Extraction

Once the column boundaries are determined using vertical green lines, the next step is to extract the text from each table cell using Optical Character Recognition (OCR). The following steps outline the complete extraction pipeline:

1. **Contour Detection:**
   The preprocessed image is analyzed using cv2.findContours() to detect rectangular regions that represent chunks of text or content.

2. **Sorting of Contours:**
   All detected contours are sorted in a top-to-bottom order using a helper function. This ensures that the reading order matches the layout of the table.

3. **Row Grouping:**
   Boxes (contours) that are close to each other in the vertical direction are grouped into the same row. A new row is created when a contour is far enough from the previous one vertically.

4. **Column Alignment:**
   For each row, the center positions of boxes are matched against the detected vertical column lines to determine the correct column index of the cell.

5. **Text Extraction with Tesseract OCR:**
   Each cell image is cropped and further cleaned using morphological operations like dilation and erosion to improve OCR performance. The cleaned image is then passed to Tesseract OCR to extract the actual text.

6. **Data Storage in Tabular Format:**
   All recognized cell texts are arranged into a 2D list corresponding to the table structure and saved using the pandas.DataFrame() function. Finally, the output is exported as a CSV file for further use.

Dataset cleaned:

	DESCRIPTION	UNIT PRICE	QTY	TOTAL
0	brand consultation	100.0	4	$100
1	logo design	100.0	1	$100
2	website design	100.0	1	$100
3	social media template	100.0	1	$100
4	brand manual	100.0	4	$100

Step	Operation	Purpose	Why It Matters
Column Detection	**Dilation**	Connects vertically aligned text	Allows detection of column "bands"
Column Detection	**Contours**	Finds regions of interest	Extracts x-coordinates of columns
Column Detection	**Clustering**	Groups close x-values	Reduces false/duplicate lines
OCR Preprocessing	**Dilation + Erosion**	Enhances + cleans character regions	Boosts OCR confidence and reduces noise

METRICS:

Metric	Score
Table Mask IoU	84.5%
Column Mask IoU	43.1%(discarded)
Column Clustering F1 score	91.2%
OCR Confidence	87.6%

REFERENCES

[1] Anand, A. et al., “TC-OCR: TableCraft OCR,” arXiv:2404.10305
[2] Paliwal, S. et al., “TableNet,” arXiv:2001.01469
[3] Prasad, D. et al., “CascadeTabNet,” arXiv:2004.12629
[4] Agarwal, M. et al., “CDeC-Net,” arXiv:2008.10831
[5] Fischer, P. et al., “Multi-Type-TD-TSR,” arXiv:2105.11021

5.0 Conclusion

This work presents an optimized end-to-end approach for table extraction that significantly reduces computational complexity while maintaining high accuracy. By leveraging lightweight architectures like MobileNetV2/EfficientNet-lite, employing knowledge distillation, and integrating shared feature extraction with minimal decoder heads, our method enhances inference speed and efficiency. Additionally, ROI pooling and morphological operations streamline preprocessing and post-processing, reducing unnecessary computations.

This hybrid approach proved effective in extracting tabular data from scanned documents, especially invoices. By abandoning the noisy deep learning column mask and switching to line-based detection, the pipeline achieved high structural accuracy and improved OCR reliability.

Limitations remain in detecting merged cells or complex nested tables. Future work can incorporate transformer-based layout models like LayoutLMv3.