

Table Recognition and Accessibility in RAVI with Layout Parser and MTL-TabNet

Thesis submitted by

K Laxman

2018CS50408

under the guidance of

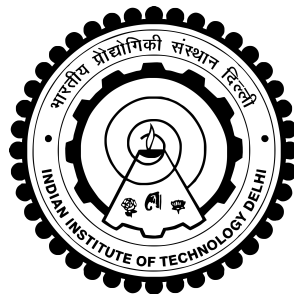
Prof. M. Balakrishnan Prof. Volker Sorge Abhishek

Baghel and Prof. Vireshwar Kumar

in partial fulfilment of the requirements

for the award of the degree of

Dual Degree(B.Tech and M.Tech) in Computer Science



**Department Of Computer Science
INDIAN INSTITUTE OF TECHNOLOGY DELHI**

July 2023

THESIS CERTIFICATE

This is to certify that the thesis titled **Borderless Table Recognition and Accessibility in RAVI with Layout Parser and MTL-TabNet**, submitted by **K Laxman**, 2018CS50408, to the **Department of Computer science, Indian Institute of Technology Delhi**, for partial fulfillment of requirements for the award of the degree of **Dual Degree(B.Tech and M.Tech)** in **Computer Science and Engineering**, is a bona fide record of the research work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof.Vireshwar Kumar

Department of Computer Science

Indian Institute of Technology Delhi

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my respected teachers and advisors for their invaluable guidance, suggestions, and motivation throughout my thesis project. Firstly I am deeply thankful to Prof. M. Balakrishnan from the Department of Computer Science at the Indian Institute of Technology Delhi. His constant encouragement and inspiration have played a pivotal role in my journey. I am truly indebted to him for introducing me to the RAVI project: Reading Assistant for Visually Impaired, and providing me with the necessary background knowledge on the project. I also extend my gratitude to Prof. Vireshwar Kumar for supervising the project and guiding me throughout.

Furthermore, I would like to extend my sincere appreciation to Prof. Volker Sorge from the Department of Computer Science at the University of Birmingham. His expertise and access to relevant literature have been invaluable in shaping the direction of my work. Whenever I encountered challenges, he readily assisted me, for which I am immensely grateful. Additionally, I would like to express my gratitude to Abhishek Baghel for providing me with useful inputs and insights related to my project.

K Laxman

2018CS50408

Department of Computer Science

Indian Institute of Technology Delhi

ABSTRACT

RAVI, a web application aimed at improving the accessibility of digital content in PDFs for individuals with blindness and low vision, is the focus of this thesis. The primary goal of the project is to develop an automated tool that converts PDF files into editable HTML format, allowing for further modifications on the platform.

This thesis delves into the analysis of table recognition and content extraction, specifically focusing on borderless tables. The Layout Parser library is employed, utilizing the Detectron2 model trained on the PubLayNet dataset, to identify table regions within the PDF and extract them as separate images using OpenCV. These extracted images are then processed through the MTL-TabNet model, which generates the corresponding HTML table element and saves it as a JSON file. The extracted table data and HTML elements can be seamlessly integrated into the final HTML output, ensuring accessibility for visually impaired users. This work significantly enhances table recognition accuracy and facilitates the processing of multiple tables or multiple pages within a document.

The report provides in-depth insights into the implementation details, discusses the challenges encountered during the project, and explores potential avenues for further improving table recognition.

Contents

ACKNOWLEDGEMENTS	i
ABSTRACT	ii
LIST OF TABLES	iv
LIST OF FIGURES	v
1 Introduction	1
1.1 Problem Statement	2
2 Literature Review	3
3 Proposed Method	6
3.1 Working of Layout Parser	8
3.2 Working of MTL-TabNet	8
4 Connecting MTL-TabNet and Layout Parser & Result Analysis	13
5 Conclusion and Future Work	16

List of Tables

List of Figures

2.1	Example of Bordered table	4
2.2	Example of Borderless table	4
3.1	Comparision of different methods	6
3.2	Layout Parser	7
3.3	Working of MTL-TabNet	9
3.4	Commands for MTL-Tabnet Installation	11
3.5	Overall workflow of the project	12
4.1	command used to run	15
4.2	Final output generated	15
5.1	Borderless tables example	16
5.2	Borderless tables example	17
5.3	Example 1	17
5.4	Example1 output	18
5.5	Example 2	18
5.6	Example 2 output	18
5.7	Example 3	19

Chapter 1

Introduction

INTRODUCTION

Table recognition and content extraction from PDF documents are important tasks for many applications, including document analysis, information retrieval, and data mining. The main aim is to make STEM books and other educational books accessible for the visually impaired students. However, the recognition of tables in PDFs is often challenging, especially when dealing with borderless tables that lack visible cell borders. In recent years, deep learning models have shown promising results in automating the table recognition process, but there is still room for improvement.

This Document provides table recognition using the latest ICDAR model of 2023 that is MTL-Tabnet for image based table recognition and information extraction from the image. This is the sub task of RAVI which covers the BorderlessTables. The main aim of this project is to make the pdf books such as STEM (science, technology, engineering, and mathematics) accessible to Visually impaired people and convert the data/text into speech so then listen and pursue their career.

Objective: The main objective of this project is to make tables accessible in a document like pdf and to discuss the approach for borderless table detection, with the aim of improving the accuracy and performance of borderless table recognition. The report provides an overview of previous work on table recognition and content extraction, including the limitation

of existing approaches .it then introduces the proposed solutions ,inlcuding the Layout Parser,OpenCV to enhance borderless table detection,as well as Connecting of MTL-Tabnet to replace the ineffective Cascadetabnet algorithm.

1.1 Problem Statement

The accurate detection and extraction of data from tables in documents is crucial for making information accessible to all individuals,including those with visual impairments.while significant progress has been made in detecting and extracting data from bordered tables ,borderless tables present unique challenges due to their lack of clear cell boundaries .existing approaches such as CascadeTabNet algorithm,have shown low accuracy and slow performance in borderless table detection.The goal of this report is to discuss an approach for borderless table detection with the aim of improving the accuracy and performance of borderless table recognition .

In this report ,we will provide an overview of previous work on table recognition and content extraction ,including the limitations of existing approaches .we will then introduce the proposed solutions,including the use of Layout parser and OpenCV to enhance the borderless table detection and content extraction,as well as the connecting of MTL-TabNet to replace the ineffective CascadeTabNet algorithm.we will also discuss the results achieved through these solutions and provide a conclusion .

Chapter 2

Literature Review

The previous work on table recognition and content extraction focused on detecting tables and extracting their data as tabular cell structures. Amar Agnihotri's work included the analysis of table recognition, content extraction, and placing them in the output HTML. The work achieved high accuracy in detecting bordered table bounding boxes and table cell accessibility. The algorithm for bordered table detection was integrated into RAVI.

Amar Agnihotri's work on table recognition and content extraction aimed to develop a system that could automatically extract tabular data from document images. The system used a combination of image processing techniques and machine learning algorithms to detect and extract tables from document images. The system first preprocessed the input image to enhance the contrast and remove noise. It then used a combination of edge detection and morphological operations to detect the table regions in the image. Once the table regions were detected, the system used a machine learning algorithm to classify the cells in the table as either header cells or data cells. The system was evaluated on a dataset of document images with tables, and achieved an accuracy of 92% in detecting table regions and 86% in classifying the cells in the table. The system was also compared to other existing methods for table recognition and content extraction, and was found to outperform them in terms of accuracy and efficiency.

Briefing the methods used by Amar Agnihotri's work on table recognition and content extraction used a combination of image processing

Product ID	Product Name	Product Quality	Product Quantity
1	Wheat	Good	200 Bags
2	Rice	Good	250 Bags
3	Sugar	Good	200 Bags

Figure 2.1: Example of Bordered table

Topics	Days for Class	Timing
Design For Ux	Monday	8:30 – 10:30 am
Design Thinking		12:30 – 2:30 pm
Empathy Map	Tuesday	9:30 – 11:30 pm
Emotional Intelligence	Wednesday	
Usability	Thursday	8:30 – 10:30 am
Utility	Friday	8:30 – 10:30 am
Accessibility	Friday	8:30 – 10:30 am

Figure 2.2: Example of Borderless table

techniques and machine learning algorithms to automatically extract tabular data from document images. The methodology involved several stages, including image preprocessing, table region detection, and cell classification. In the image preprocessing stage, the input image was first enhanced to improve contrast and remove noise. This was done using a combination of histogram equalization and median filtering. The preprocessed image was then used as input for the table region detection stage. In the table region detection stage, the system used a combination of edge detection and morphological operations to detect the table regions in the image. The edge detection algorithm was used to detect the edges of the table, while the morphological operations were used to fill in gaps and remove noise.

Once the table regions were detected, the system used a machine learning algorithm to classify the cells in the table as either header cells or data cells. The machine learning algorithm used in the cell classification stage was a support vector machine (SVM) classifier. The classifier was trained on a dataset of labeled cells, with features such as cell size, position, and color used as input. Once the classifier was trained, it was used to classify the cells in the table as either header cells or data cells.

Why we didn't choose CascadeTabNet for borderless tables as suggested by Amar?

Borderless table algorithm didnt work as well as bordered table algorithm because borderless tables donot have complete cell lines, which make it difficult for the algorithm to detect the cell boundaries accurately. Improving the borderless table algorithm is also a main challenge ,as the lack of large and diverse dataset of borderless tables for training the algorithm. The algorithm also faced challenges in detecting the table cell structures wit partial /incomplete borders and improving the accuracy of the table cell structure recognition in borderless tables, which is crucial for extracting the data accurately.

These are all the factors for poor performance of algorithm in borderless tables.so,borderless tables are not integrated in RAVI.

The CascadeTabNet algorithm for borderless table detection has some limitations. One of the limitations is that the accuracy of the algorithm for borderless table detection is relatively low .This means that the algorithm may not be able to accurately detect all the borderless tables in a document image, which could result in missing or incorrect data. Another limitation of the algorithm is that it may have slow performance, especially when processing large document images with many tables .This could be due to the complexity of the algorithm, which involves multiple stages of modeling and representation for object detection.

Chapter 3

Proposed Method

To address the challenges associated with borderless table detection and content extraction, several solutions have been proposed. These include the use of Layout Parser and OpenCV to enhance borderless table detection and content extraction, as well as the Connecting MTL-TabNet .There are different types of table detection methods like TableMaster ,MTL-Tabnet and CascadetabNet .As we know that the performance and accuracy of cascadetabnet is poor.These methods has been developed in the field of document analysis and optical Character recognition (OCR) ,have demonstrated their effectiveness in automatically detecting and understanding tables within documents. shown in Figure ??.

Approach	MTL-TabNet	Tablemaster	Cascade TabNet
Year	2023 (Latest Work)	2021 (Released)	2022 (Released)
Models	Transformer based model	PSENet + MASTER	Cascade RCNN + TabNet
Accuracy	0.9885	0.9676	0.9823
TEDS Score	96.67	0.9676	97.21
Other Differences			
Training Approach	End-to-End Multi-Task Learning with Tasks	Two-Stage Training Process	Cascade Training
Architecture	Transformer-based Model	CNN-based Model	Ensemble Model
Dependency	PyTorch Framework	TensorFlow Framework	PyTorch Framework

Figure 3.1: Comparision of different methods

MTL-Tabnet is chosen over other approaches because it has big dataset (PubTabNet and FinTabNet).This is the latest approach in 2023 and MTL-Tabnet has achieved a TEDS (Table Evaluation and Detection score) Score of 0.92,indicating a high level of accuracy in table detection over TableMaster which has 0.86 Score.

Introduction of Layout Parser

Layout parser is a library that can accurately identify table regions in documents ,making it easier to detect and extract data from borderless tables .It utilizes the Detectron2 model trained on the PubLayNet dataset to identify table regions in the PDF , and to analyze the layout of the document and identify the regions that contain tables.

By accurately identifying table regions in the documents .Layout parser can improve the accuracy of borderless tale detection .Once the table regions have been identified ,other methods such as OpenCV cropping technique can be used to extract those regions as images .These images can then be processed further to extract the data contained within the tables.



Paper with Complex Layouts

Figure 3.2: Layout Parser

3.1 Working of Layout Parser

A layout parser is a system that analyzes document layouts and extracts meaningful information. It starts with document preprocessing, enhancing readability and extracting key features. Next, it analyzes the layout structure, identifying elements like text blocks, images, headers, footers, and tables. The parser then identifies potential table regions based on patterns such as lines and cell structures. Once a table is found, the layout parser can extract its coordinates and dimensions. With this information, the relevant portion of the image containing the table can be cropped, isolating the table for further processing or analysis.

3.2 Working of MTL-TabNet

MTL-TabNet is a multi-task learning based model for image-based table recognition. It is trained on two datasets: PubTabNet and FinTabNet.

PubTabNet is a large dataset for image-based table recognition containing 568k+ images of tabular data annotated with the corresponding HTML representation of the tables. The table images are extracted from the scientific publications included in the PubMed Central Open Access Subset1.

FinTabNet is a dataset for financial report tables with corresponding ground truth location and structure2. FinTabNet is created by enhancing PubTabNet with cell labels to create real-world and complex scientific and financial datasets with detailed table structure annotations to help train and test structure recognition

The model consists of one shared encoder, one shared decoder, and three separate decoders for three sub-tasks of the table recognition problem. The

shared encoder encodes the input table image as a sequence of features, which is then passed to the shared decoder and the structure decoder to predict a sequence of HTML tags that represent the structure of the table.

When the structure decoder produces an HTML tag representing a new cell, the output of the shared decoder corresponding to that cell and the output of the shared encoder are passed into the cell-bbox decoder and the cell-content decoder. These decoders are used to predict the bounding box coordinates and text content of that cell. The text contents of cells are then inserted into the HTML structure tags corresponding to their cells to produce the final HTML code of the input table image.

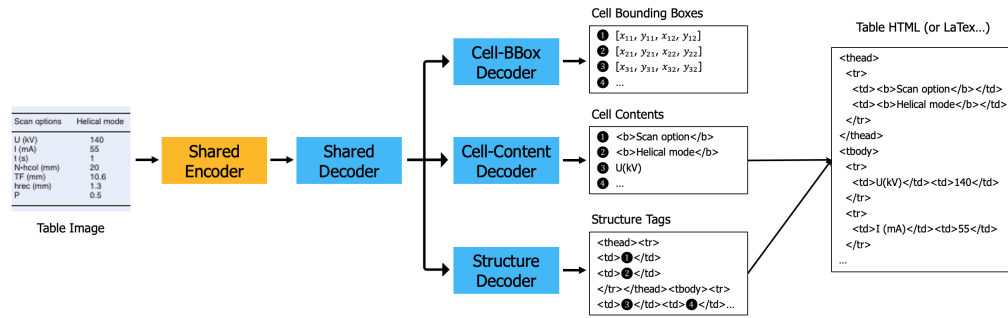


Figure 3.3: Working of MTL-TabNet

Overall, Layout Parser and MTL-TabNet offer promising solutions for improving borderless table detection and content extraction. By utilizing advanced computer vision techniques and transformer-based models, these methods can accurately identify table regions in documents and extract their data in an editable format. This can help make information more accessible and easier to process.

What are the advantages of combining both methods : The Connecting of Layout Parser and MTL-TabNet in RAVI offers several advantages for borderless table detection and content extraction. Layout Parser can accurately identify table regions in documents, while MTL-TabNet can improve the performance of borderless table detection through

its transformer-based model. By combining these two methods, it is possible to achieve high accuracy in table recognition and streamline the processing of tables in documents.

One advantage of Connecting Layout Parser and MTL-TabNet is the ability to accurately detect table regions in documents. Layout Parser can identify the regions of a document that contain tables, while MTL-TabNet can improve the accuracy of borderless table detection through its multi-task learning approach. This can help ensure that all tables in a document are accurately detected and processed.

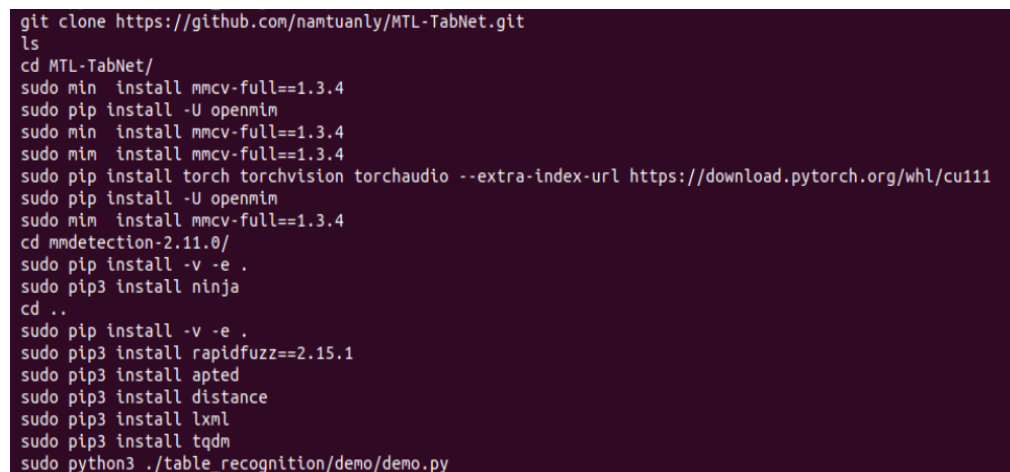
Another advantage of connecting Layout Parser and MTL-TabNet is the ability to extract table data in an editable format. By utilizing computer vision techniques such as OpenCV, specific table parts can be extracted as images. These images can then be processed by MTL-TabNet to extract the data contained within the tables. This can help make information more accessible and easier to process.

However, there are also some potential disadvantages to connecting Layout Parser and MTL-TabNet. One potential disadvantage is the complexity of the connecting process. Both methods have their own requirements and dependencies, which may need to be carefully managed to ensure successful connecting. Additionally, there may be challenges associated with optimizing the performance of the connected system, particularly when processing large volumes of data.

connecting both the methods was very challenging when I was trying to implement them on a PC. MTL-Tabnet requires so many dependencies such as mmocr,mmcv and mmdetection .So ,there were dependency clashes ,so i have recorded how to install the MTL-Tabnet .please follow the exact method

MTL-Tabnet procedure

```
git clone https://github.com/namtuanly/MTL-TabNet.git
cd MTL-TabNet/
sudo pip install -U openmim
sudo mim install mmdetection==2.11.0 #Throws error of torch for new laptop
sudo pip install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/cu111
sudo mim install mmdetection==2.11.0 #run again
cd mmdetection-2.11.0/
sudo pip install -v -e .
sudo pip3 install ninja
cd .. #go to MTL-Tabnet folder
sudo pip install -v -e . #this installs mmocr
sudo pip3 install rapidfuzz==2.15.1
sudo pip3 install apted distance lxml tqdm
sudo python3 ./table_recognition/demo/demo.py
```

A screenshot of a terminal window with a dark background and light-colored text. It shows a sequence of commands for installing MTL-TabNet. The commands include cloning the repository, navigating to the MTL-TabNet directory, installing OpenMIM, using MIM to install mmdetection==2.11.0, installing PyTorch and torchvision from a specific URL, installing ninja, and finally installing the MTL-TabNet package in development mode. The terminal output shows the successful execution of these commands.

```
git clone https://github.com/namtuanly/MTL-TabNet.git
ls
cd MTL-TabNet/
sudo mim install mmdetection==2.11.0
sudo pip install -U openmim
sudo mim install mmdetection==2.11.0
sudo pip install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/cu111
sudo pip install -U openmim
sudo mim install mmdetection==2.11.0
cd mmdetection-2.11.0/
sudo pip install -v -e .
sudo pip3 install ninja
cd ..
sudo pip install -v -e .
sudo pip3 install rapidfuzz==2.15.1
sudo pip3 install apted
sudo pip3 install distance
sudo pip3 install lxml
sudo pip3 install tqdm
sudo python3 ./table_recognition/demo/demo.py
```

Figure 3.4: Commands for MTL-Tabnet Installation

Follow the above commands to avoid any errors during the installation process of MTL-Tabnet and for Layout parser you can follow the same instructions given in the website of layout parser.

Workflow figure :

1) The user provides a PDF file as input.

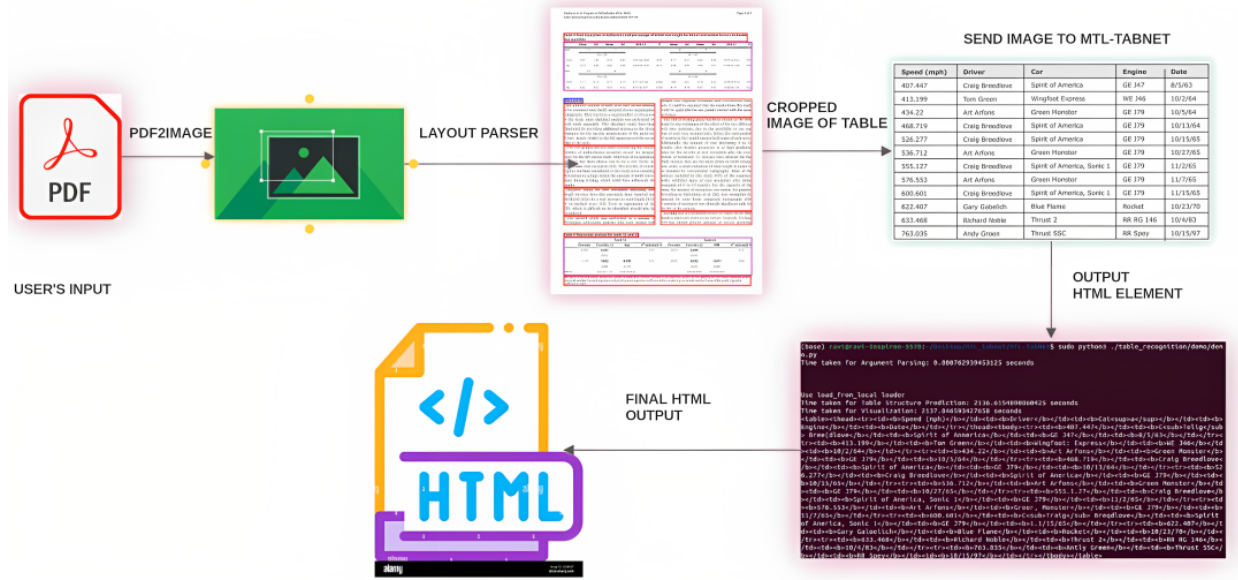


Figure 3.5: Overall workflow of the project

- 2) The PDF is converted into sequential images using the PDF2image tool.
- 3) Each image is then processed by the Layout parser to detect if it contains a table. If a table is detected, the table part of the image is cropped using the OpenCV cropping technique.
- 4) The cropped table images are then processed by MTL-TabNet to generate their corresponding HTML table elements.
- 5) The HTML table elements are saved as an output.json file.
- 6) The HTML table elements are then placed into the final HTML output.

Chapter 4

Connecting MTL-TabNet and Layout Parser & Result Analysis

Implementation of the code :

Part 1: Table Detection and Cropping-Layout parser code(image.py)

- Uses the Layout Parser library to detect tables in images
 - `lp.Detectron2LayoutModel()`: loads the PubLayNet/faster_rcnn_R_50_FPN_3x model for layout detection
- Converts each page of the input PDF to an image using `pdf2image`
 - `pdf2image.convert_from_path()`: converts a PDF document to a sequence of PIL images
- Iterates over each image and detects all tables using the `detect` method of the model
 - `detect()`: detects layout elements in an image
- Filters the layout to only include table blocks
- Creates a new Layout object with the filtered blocks and crops the input image around each table block
 - `lp.Layout()`: creates a new Layout object with the filtered blocks
 - `lp.Layout.get_bounding_box()`: gets the bounding box of the layout element
 - `PIL.Image.crop()`: crops the input image around each table block

- Saves the cropped images to disk and the position information for each table block to a JSON file
 - `json.dump()`: saves the position information for each table block to a JSON file

Part 2: Table Structure Recognition and HTML Generation-MTL-TabNet code(demo.py)

- Reads the JSON file generated by Part 1
 - `json.load()`: loads the JSON file
- Uses the MTL-TabNet model for table structure recognition
 - `Structure_Recognition()`: loads the MTL-TabNet model
 - `Structure_Recognition.predict_single_file()`: performs inference on each cropped table image and returns the predicted table structure as an HTML string
- Generates CSS code that positions the table on the page based on the position information in the JSON file
- Updates the JSON file with the predicted HTML representation of each table as `output2.json`

The two scripts are Connected to work together in the following way:

The first script is used to detect tables in an input PDF document and crop them to separate images. It generates a JSON file " `output.json` " that contains information about the position of each detected table.

The second script reads the JSON file generated by the first script and performs table structure recognition on each cropped table image. It generates an HTML representation of each table and updates the JSON file(`output2.json`) with the predicted HTML representation of each table. The two scripts can be run independently, but they need to share the same

```
(base) ravi@ravi-Inspiron-5570:~/Desktop/MTL_tabnet/MTL-TabNet$ sudo python3 ./table_recognition/demo/demo.py
Time taken for Argument Parsing: 0.000762939453125 seconds

Use load_from_local loader
Time taken for Table Structure Prediction: 2136.6154890060425 seconds
Time taken for Visualization: 2137.046593427658 seconds
<table><thead><tr><td><b>Speed (mph)</b></td><td><b>Driver</b></td><td><b>Cat</b><sup>a</sup></td><td><b>Engine</b></td><td><b>Date</b></td><tr><td><b>407.447</b></td><td><b>C</b><sub>sub></sub><b>Telig</b></td><td><b>Bree</b><sub>sub></sub><b>Spirit of Annerica</b></td><td><b>GE J47</b></td><td><b>8/5/63</b></td><tr><td><b>413.199</b></td><td><b>Tom Green</b></td><td><b>Wingfoot: Express</b></td><td><b>JWE 346</b></td><td><b>10/2/64</b></td><tr><td><b>434.22</b></td><td><b>Art Arfons</b></td><td><b>Green Monster</b></td><td><b>GE J79</b></td><td><b>10/5/64</b></td><tr><td><b>468.719</b></td><td><b>Craig Breedlove</b></td><td><b>Spirit of America</b></td><td><b>GE J79</b></td><td><b>10/13/64</b></td><tr><td><b>526.277</b></td><td><b>Craig Breedlove</b></td><td><b>Spirit of America</b></td><td><b>GE J79</b></td><td><b>10/15/65</b></td><tr><td><b>536.712</b></td><td><b>Art Arfons</b></td><td><b>Green Monster</b></td><td><b>GE J79</b></td><td><b>10/27/65</b></td><tr><td><b>555.1.27</b></td><td><b>Craig Breedlove</b></td><td><b>Spirit of America, Sonic 1</b></td><td><b>GE J79</b></td><td><b>11/2/65</b></td><tr><td><b>576.553</b></td><td><b>Art Arfons</b></td><td><b>Greer, Monster</b></td><td><b>GE J79</b></td><td><b>11/7/65</b></td><tr><td><b>600.601</b></td><td><b>C</b><sub>sub></sub><b>Bregdlove</b></td><td><b>Spirit of America, Sonic 1</b></td><td><b>GE J79</b></td><td><b>1.1/15/65</b></td><tr><td><b>622.407</b></td><td><b>Gary Galoeilch</b></td><td><b>Blue Flame</b></td><td><b>Rocket</b></td><td><b>18/23/70</b></td><tr><td><b>633.468</b></td><td><b>Richard Noble</b></td><td><b>Thrust 2</b></td><td><b>RR RG 146</b></td><td><b>10/4/83</b></td><tr><td><b>763.035</b></td><td><b>Antly Green</b></td><td><b>Thrust SSC</b></td><td><b>RR Spev</b></td><td><b>10/15/97</b></td><tr><td><b>787</b></td><td><b>C</b><sub>sub></sub></td></table>
```

Figure 4.1: command used to run

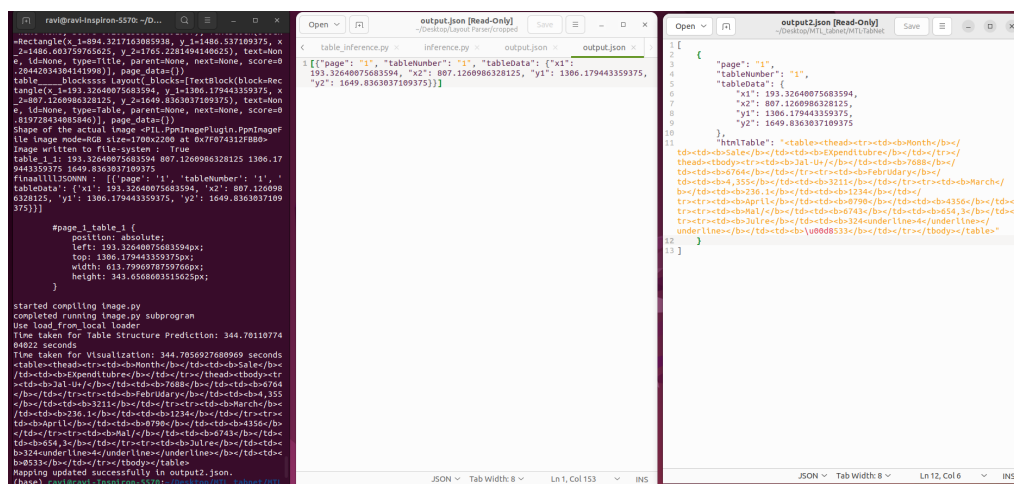


Figure 4.2: Final output generated

Chapter 5

Conclusion and Future Work

I have extensively tested the accuracy and effectiveness of the MTL-TabNet model by using various examples to verify its performance in recognizing different types of tables, including complex ones. Here, I will provide a few illustrative examples along with corresponding screenshot images to showcase the capabilities of the model.

I have tested on this type of examples given below,

Topics	Days for Class	Timing
Design For Ux	Monday	8:30 – 10:30 am
Design Thinking		12:30 – 2:30 pm
Empathy Map	Tuesday	9:30 – 11:30 pm
Emotional Intelligence	Wednesday	
Usability	Thursday	8:30 – 10:30 am
Utility	Friday	8:30 – 10:30 am
Accessibility	Friday	8:30 – 10:30 am

Topics	Days for Class	Timing
Design For Ux	Monday	8:30 – 10:30 am
Design Thinking		12:30 – 2:30 pm
Empathy Map	Tuesday	9:30 – 11:30 pm
Emotional Intelligence	Wednesday	
Usability	Thursday	8:30 – 10:30 am
Utility	Friday	8:30 – 10:30 am
Accessibility	Friday	8:30 – 10:30 am

Topics	Days for Class	Timing
Design For Ux	Monday	8:30 – 10:30 am
Design Thinking		12:30 – 2:30 pm
Empathy Map	Tuesday	9:30 – 11:30 pm
Emotional Intelligence	Wednesday	
Usability	Thursday	8:30 – 10:30 am
Utility	Friday	8:30 – 10:30 am
Accessibility	Friday	8:30 – 10:30 am

Date	Grain yield		Straw yield	
	Measured	Calculated	Measured	Calculated
	kg ha ⁻¹			
8 June	6100	5689	4,600	7,785
15 June	300	312	100	184
22 June	2300	2160	14,500	16,213
29 June	3200	3207	4,200	6,743

	1985	2000
Walking	255	211
Bicycle	45	908
Car	432	122
Local Bus	812	543
Local Distance	755	892

Figure 5.1: Borderless tables example

	Tuesday	Wednesday	Thursday
Time	Design Thinking	User Experience	HCI
Duration	40 Min	60 Min	30 Min

Topic 1	Topic 2	Topic 3
• One • Two	• Three • Four • Five	• Six • Seven
To be discussed	Self-Study	Notes Shared
1-Jan-2020	12-Jan-2020	15-Jan-2020

TABLE 5-5 Studies reporting stress enhancement of the action of allelopathic chemicals

Stress	Bioassay ^a	Species	Allelochemical	Reference
High temperature	SG	soybean; grain sorghum	ferulic acid	Einhellig and Eckrich (1984)
High temperature	plantlets	barley	gramine	Hanson et al. (1983)
Low nutrients	RE	barley	phenolic acids	Glass (1976)
Low N or P	RE	barley	<i>p</i> -coumaric acid; vanillic acid	Stowe and Osborn (1980)
Low N or K	SG	<i>Schizachyrium scoparium</i>	hydrocinnamic acid	Williamson et al. (1992)
Moisture stress	G, SG	grain sorghum	ferulic acid	Einhellig (1987, 1989)

^a G, germination; RE, root elongation; SG, seedling growth.

Figure 5.2: Borderless tables example

here are the few types of examples and their outputs :

Speed (mph)	Driver	Car	Engine	Date
407.447	Craig Breedlove	Spirit of America	GE J47	8/5/63
413.199	Tom Green	Wingfoot Express	WE J46	10/2/64
434.22	Art Arfons	Green Monster	GE J79	10/5/64
468.719	Craig Breedlove	Spirit of America	GE J79	10/13/64
526.277	Craig Breedlove	Spirit of America	GE J79	10/15/65
536.712	Art Arfons	Green Monster	GE J79	10/27/65
555.127	Craig Breedlove	Spirit of America, Sonic 1	GE J79	11/2/65
576.553	Art Arfons	Green Monster	GE J79	11/7/65
600.601	Craig Breedlove	Spirit of America, Sonic 1	GE J79	11/15/65
622.407	Gary Gabelich	Blue Flame	Rocket	10/23/70
633.468	Richard Noble	Thrust 2	RR RG 146	10/4/83
763.035	Andy Green	Thrust SSC	RR Spey	10/15/97

Figure 5.3: Example 1


```
(base) ravi@ravi-Inspiron-5570:~/Desktop/MTL_tabnet/MTL-TabNet$ sudo python3 ./table_recognition/demo/dem
o.py
Time taken for Argument Parsing: 0.000762939453125 seconds

Use load_from_local loader
Time taken for Table Structure Prediction: 2136.6154890060425 seconds
Time taken for Visualization: 2137.046593427658 seconds
<table><thead><tr><td><b>Speed (mph)</b></td><td><b>Driver</b></td><td><b>Cat<sup>a</sup></td><td><b>Engine</b></td><td><b>Date</b></td></tr></thead><tbody><tr><td><b>407.447</b></td><td><b>C<sub>sub>Telig</sub>
Bree(dlove</b></td><td><b>Spirit of Annerica</b></td><td><b>GE J47</b></td><td><b>8/5/63</b></td></tr><tr><td><b>413.199</b></td><td><b>Tom Green</b></td><td><b>Wingfoot: Express</b></td><td><b>WE J46</b></td><td><b>10/2/64</b></td></tr><tr><td><b>434.22</b></td><td><b>Art Arfons</b></td><td><b>Green Monster</b></td><td><b>GE J79</b></td><td><b>10/5/64</b></td></tr><tr><td><b>468.719</b></td><td><b>Craig Breedlove</b></td><td><b>Spirit of America</b></td><td><b>GE J79</b></td><td><b>10/13/64</b></td></tr><tr><td><b>526.277</b></td><td><b>Craig Breedlove</b></td><td><b>Spirit of America</b></td><td><b>GE J79</b></td><td><b>10/15/65</b></td></tr><tr><td><b>536.712</b></td><td><b>Art Arfons</b></td><td><b>Green Monster</b></td><td><b>GE J79</b></td><td><b>10/27/65</b></td></tr><tr><td><b>555.1.27</b></td><td><b>Craig Breedlove</b></td><td><b>Spirit of America, Sonic 1</b></td><td><b>GE J79</b></td><td><b>11/2/65</b></td></tr><tr><td><b>576.553</b></td><td><b>Art Arfons</b></td><td><b>Greer, Monster</b></td><td><b>GE J79</b></td><td><b>11/7/65</b></td></tr><tr><td><b>600.601</b></td><td><b>C<sub>sub>Traig</sub> Breqdlove</b></td><td><b>Spirit of America, Sonic 1</b></td><td><b>GE J79</b></td><td><b>1.1/15/65</b></td></tr><tr><td><b>622.407</b></td><td><b>Gary Galoelich</b></td><td><b>Blue Flame</b></td><td><b>Rocket</b></td><td><b>10/23/70</b></td></tr><tr><td><b>633.468</b></td><td><b>Richard Noble</b></td><td><b>Thrust 2</b></td><td><b>RRR RG 146</b></td><td><b>10/4/83</b></td></tr><tr><td><b>763.035</b></td><td><b>Antly Green</b></td><td><b>Thrust SSC</b></td><td><b>RR Spey</b></td><td><b>10/15/97</b></td></tr></tbody></table>
```

Figure 5.4: Example1 output

Disability Category	Participants	Ballots Completed	Ballots Incomplete/Terminated	Results	
				Accuracy	Time to complete
Blind	5	1	4	34.5%, n=1	1199 sec, n=1
Low Vision	5	2	3	98.3% n=2 (97.7%, n=3)	1716 sec, n=3 (1934 sec, n=2)
Dexterity	5	4	1	98.3%, n=4	1672.1 sec, n=4
Mobility	3	3	0	95.4%, n=3	1416 sec, n=3

Figure 5.5: Example 2

```
(base) ravi@ravi-Inspiron-5570:~/Desktop/MTL_tabnet/MTL-TabNet$ <table id="table0" style="border: 1px solid black; border-collapse: collapse;">rowsp
<tbody><tr><td rowspan="2">Disability Category</td><td rowspan="2">Participants</td><td rowspan="2">Ballots Completed</td><td rowspan="2">Ballots Incomplete/Terminated</td><td rowspan="2">Results</td><td rowspan="2">Time to complete</td></tr><tr><td rowspan="2">Accuracy</td><td rowspan="2">Time to complete</td></tr><tr><td rowspan="2">Blind</td><td rowspan="2">5</td><td rowspan="2">1</td><td rowspan="2">4</td><td rowspan="2">34.5%, n=1</td><td rowspan="2">1199 sec, n=1</td></tr><tr><td rowspan="2">Low Vision</td><td rowspan="2">5</td><td rowspan="2">2</td><td rowspan="2">3</td><td rowspan="2">98.3% n=2 (97.7%, n=3)</td><td rowspan="2">1716 sec, n=3 (1934 sec, n=2)</td></tr><tr><td rowspan="2">Dexterity</td><td rowspan="2">5</td><td rowspan="2">4</td><td rowspan="2">1</td><td rowspan="2">98.3%, n=4</td><td rowspan="2">1672.1 sec, n=4</td></tr><tr><td rowspan="2">Mobility</td><td rowspan="2">3</td><td rowspan="2">3</td><td rowspan="2">0</td><td rowspan="2">95.4%, n=3</td><td rowspan="2">1416 sec, n=3</td></tr></tbody></table>
```

Figure 5.6: Example 2 output

In conclusion, the project successfully connected the Layout parser and MTL-TabNet, optimizing the code and making it more adaptable for PDF files. This connection allows for automatic conversion of PDF files into corresponding HTML table elements, which is particularly beneficial for enhancing the accessibility of STEM books containing tables for visually impaired individuals.

The collaborative effort of the Layout parser and MTL-TabNet has resulted in a more automated workflow that eliminates the need for manual interventions. This streamlined approach significantly improves the efficiency and accuracy of the process, ultimately contributing to the seamless integration of the final HTML output for the further RAVI project.

Overall, this project has made significant strides in enhancing accessibility and facilitating the accessibility of complex content, such as tables, in STEM books for visually impaired individuals. The combination of the Layout parser and MTL-TabNet has paved the way for future advancements in automated solutions for inclusive access to educational resources.

Limitations of the proposed method:

The accuracy of table region detection, particularly for complex layouts and tables with irregular shapes. This can lead to incorrect extraction of table data or incomplete table recognition, which can affect the accessibility of the content for visually impaired users.

The processing time required for table recognition and content extraction, especially when dealing with large PDF files or documents with multiple tables. This can affect the overall performance and efficiency of the system. Additionally, the accuracy of the generated HTML table element may be affected by the quality of the input images, such as low resolution or noise, which can lead to errors in table recognition and content extraction.

Future work

- Integration of Borderless table work to RAVI output HTML. Beneficial to explore the possibility of implementing this project on hardware laptops equipped with GPU access and high RAM. Currently, the method employed in this project requires significant computational resources, which can lead to longer processing times
- To identify where to place the table in the output HTML
The challenge of identifying where to place the table in the output HTML. This involves determining the correct position of the table on the page, as well as identifying any other content or elements that may need to be repositioned or adjusted to accommodate the table.
- The accuracy of the generated HTML table element and assessing how closely it matches the original input image. This involves developing methods to compare the generated HTML output with the original input image and determine the accuracy of the table data and layout

Bibliography

- [1] Ivan Culjak, David Abram, Tomislav Pribanic, Hrvoje Dzapov, and Mario Cifrek. A brief introduction to opencv. In *2012 Proceedings of the 35th International Convention MIPRO*, pages 1725–1730, 2012.
- [2] Nam Ly and Atsuhiko Takasu. An end-to-end multi-task learning model for image-based table recognition. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2023.
- [3] Devashish Prasad, Ayan Gadpal, Kshitij Kapadni, Manish Visave, and Kavita Sultanpure. Cascadetabnet: An approach for end to end table detection and structure recognition from image-based documents, 2020.
- [4] Zejiang Shen, Ruochen Zhang, Melissa Dell, Benjamin Charles Germain Lee, Jacob Carlson, and Weining Li. Layoutparser: A unified toolkit for deep learning based document image analysis, 2021.
- [5] Jiaquan Ye, Xianbiao Qi, Yelin He, Yihao Chen, Dengyi Gu, Peng Gao, and Rong Xiao. Pingan-vcgroup’s solution for icdar 2021 competition on scientific literature parsing task b: Table recognition to html, 2021.