Artificial Neural Networks-Driven High Precision Tabular Information Extraction from Datasheets

by

Johan Fernandes

A thesis
presented to the University of Ottawa
in fulfillment of the
thesis requirement for the degree of
Master of Computer Science Concentration in Applied Artificial Intelligence

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Global organizations have adopted Industry 4.0 practices to stay viable through the information shared through billions of digital documents. The information in such documents is vital to the daily functioning of such organizations. Most critical information is laid out in tabular format in order to provide the information in a concise manner. Extracting this critical data and providing access to the latest information can help institutions to make evidence based and data driven decisions. Assembling such data for analysis can further enable organizations to automate certain processes such as manufacturing. A generalized solution for table text extraction would have to handle the variations in the page content and table layouts in order to accurately extract the text. We hypothesize that a table text extraction pipeline can extract this data in three stages. The first stage would involve identifying the images that contain tables and detecting the table region. The second stage would consider the detected table region and detect the rows and columns of the table. The last stage would involve extracting the text from the cell locations generated by the intersecting lines of the detected rows and columns.

For first stage of the pipeline, we propose TableDet: a deep learning (artificial neural network) based methodology to solve table detection and table image classification in datasheet (document) images in a single inference. TableDet utilizes a Cascade R-CNN architecture with Complete IOU (CIOU) loss at each box head and a deformable convolution backbone to capture the variations of tables that appear at multiple scales and orientations. It also detects text and figures to enhance its table detection performance. We demonstrate the effectiveness of training TableDet with a dual-step transfer learning process and fine-tuning it with Table Aware Cutout (TAC) augmented images. TableDet achieves the highest F1 score for table detection against state-of-the-art solutions on ICDAR 2013 (complete set), ICDAR 2017 (test set) and ICDAR 2019 (test set) with 100%, 99.3% and 95.1% respectively. We show that the enhanced table detection performance can be utilized to address the table image classification task with the addition of a classification head which comprises of 3 conditions. For the table image classification task TableDet achieves 100% recall and above 92% precision on three test sets. These classification results indicate that all images with tables along with a significantly reduced number of images without tables would be promoted to the next stage of the table text extraction pipeline.

For the second stage we propose TableStrDet, a deep learning (artificial neural network) based approach to recognize the structure of the detected tables regions from stage 1 by detecting and classifying rows and columns. TableStrDet comprises of two Cascade R-CNN architectures each with a deformable backbone and Complete IOU loss to improve their detection performance. One architecture detects and classifies columns as regular columns

(column without a merged cell) and irregular columns (group of regular columns that share a merged cell). The second architecture detects and classifies rows as regular rows (row without a merged cell) and irregular rows (group of regular rows that share a merged cell). Both architectures work in parallel to provide the results in a single inference. We show that utilizing TableStrDet to detect four classes of objects enhances the quality of table structure detection by capturing table contents that may or may not have hierarchical layouts on two public test sets. Under the TabStructDB test set we achieve 72.7% and 78.5% weighted average F1 score for rows and columns respectively. On the ICDAR 2013 test set we achieve 90.5% and 89.6% weighted average F1 score for rows and columns respectively. Furthermore, we show that TableStrDet has a higher generalization potential on the available datasets.

Acknowledgements

I would like to express my gratitude towards my supervisor Dr. Burak Kantarci for his limitless enthusiasm, support and guidance. Thanks to Dr. Burak, I can now say that I have learnt what it truly means to perform research and how to keep pressing forward to improve the quality of my work. Working with him I was inspired to develop an unwavering determination, to keep an open mind and to strive for my most ambitious goals. Without his support I do not believe that this thesis would have been completed nor would it's quality have been improved. I really appreciate the valuable feedback that he provided during my entire time at the NEXTCON lab. I believe his feedback has helped me boost my interest in the world of research and the applications of Artificial Intelligence.

I would also like to thank JiChu Jiang, Yakup Akkaya and Dr. Murat Simsek for their constructive and valuable feedback. It was a pleasure to work and learn from them and I am grateful to have the opportunity to work with such a great team. I would also like to thank other members of the NEXTCON lab who supported my research and from whom I have learnt a great deal. Thank you Jinxin Liu, Nima Taherifard, Zhiyan Chen, Ahmed Omara, Nahid Parvaresh and Yu Shen.

I would also like to thank Dr. Shahzad Khan and the members of Lytica Inc. who supported this project. Thank you for providing the data and subject matter expertise that helped me and my colleagues to develop this project. I would like to thank the MITACS Accelerate IT14836 internship program that helped us students connect with members at Lytica Inc. with whom we have built this project that we hope will benefit the field of document analysis in the near future. I would also like to thank Dr. Burak and Dr. Shahzad for securing the Compute Canada GPU resources under SOSCIP Project 3-040. Without these resources I believe the exceptional work that we were able to achieve would have been unimaginable. I sincerely thank you both for securing these resources to help us apply, learn and develop our work.

I would finally like to thank my parents and my brother. Thanks to their constant support and encouragement I was able to achieve so much in such short span in Canada. I am truly grateful for the opportunity they have given me to not only study in Canada but to pursue a career in the field of Artificial Intelligence. I would also like to thank my close relatives who live in the US and Canada for giving me the support to help me settle in Canada. I hope I've made my family proud with this work and I hope to continue making them proud in the future.

Table of Contents

List of Tables

List of Figures

Publications of the Candidate During MCS AAI Studies

Publications during the Master's study (Journal Articles):

- **J. Fernandes**, M. Simsek, B. Kantarci, and S. Khan, "TableDet: An End-to-End Deep Learning Approach for Table Detection and Table Image Classification in Data Sheet Images" in *Elsevier Neurocomputing* (Published)
- **J. Fernandes**, M. Simsek, B. Kantarci, and S. Khan, "TableStrDet: Framework for Table Structure Detection in Data Sheet Images" in *International Journal on Document Analysis and Recognition (IJDAR)* (Under Review)
- J. Liu, M. Nogueira, **J. Fernandes** and B. Kantarci, "Adversarial Machine Learning: A Multi-Layer Review of the State-of-the-Art and Challenges for Wireless and Mobile Systems," in IEEE Communications Surveys Tutorials, doi: 10.1109/COMST.2021.3136132.

Chapter 1

Introduction

The introduction of Fifth Generation (5G) networks has made it possible to share large quantities of information in various formats between different entities. It provides transfer rates that are 10-100 times faster than the current and more prominently available Fourth Generation (4G) networks [1]. In addition, the reliability offered by 5G networks has prompted organizations to work towards the possibility of automating processes such as manufacturing. The notion of automating processes with the help of 5G networks is supported by the implementation of Industry 4.0 (I4.0) as industries shift from mass manufacturing to custom manufacturing [1]. I4.0 requires technologies such as Big Data Analytics, Artificial Intelligence, Cloud computing and Internet of Things (IoT) to ensure automation of processes such as manufacturing [2]. These technologies require robust networks such as 5G in order to provide real time delivery of information that ensures the automated process is a success. Furthermore, technologies such as IoT and Cloud computing provide services to collect and store large amounts of data which is made available to every member (human) of an corporation or enterprise. This enables the members to make analytical based critical decisions with the help of the available data [2].

The practices of I4.0 are not limited to a single corporation but can be expanded to supply chains as well. A supply chain consist of network of organizations such as manufacturers and product designers. As organizations of supply chains now exist across the globe at different locations with the implementation I4.0 practices it becomes more necessary to have the latest information available to all members of these organizations [3] in real time. Having access to such information can aide with decisions such as the production of which product should be increased or decreased or which organizations can handle what amount of production [3]. However, each organization on the supply chain produces information in a format that can be easily interpreted by members of that or

similar organizations. With no general structured layout of information that is utilized by all organizations, sharing information between them in real time requires some additional processing [4]. Organizations share information through reports and documents which are available either electronically in PDF or Docx format or in paper format. These reports and documents which we refer to as datasheets can contain information such as product specifications, product cost etc. The important data in such documents is placed in tabular format to provide it in a concise format. Utilizing tabular format to provide important data is not limited to organizations of supply chains. It can also be found in documents and reports by governments and even in scientific reports.

Extracting and assembling the information from such tables for analysis can help organizations to develop sophisticated knowledge management systems. This in turn can support efficient automation as well as evidence and data-driven decision-making [5]. Thus, there has been a growing interest in the field of document analysis that specifically deals with identification of tables and extraction of text from the tables [6]. There have been proposed solutions that extract text from tables that appear in PDF files [7]. However, the rules set by extraction processes for PDF files are limited to function only on that type of documents or reports. Thus, the recent trend of table text extraction now focuses on images of document pages rather than PDF files. Competitions such as Table Detection and Recognition at ICDAR 2019 [8] display this increasing trend towards addressing table detection and table structure recognition in images rather than digitally born PDF files. In addition, using images helps broaden the scope of document analysis from digital-born document pages that are converted to images to scanned document pages.

The advancements in field of Computer Vision brought on by the use of Artificial Neural Networks (Deep Learning) has further led to an increased interest to apply the latest developments for document analysis. Deep Learning solutions (models) have achieved excellent performances for tasks such as image classification [9] and object detection [10] on large datasets such as COCO and ImageNet. The major advantage of utilizing Deep Learning solutions is its ability to generalize. For instance, models such as Cascade R-CNN [11] have achieved excellent detection performance in detecting 80+ naturally occurring objects in the COCO dataset. These solutions can be used to solve object detection tasks on images that do not contain naturally occurring objects as well as shown in [10]. Thus, in this work we propose utilizing a deep learning-based approach to analyze tables in document or datasheet images.

We hypothesize that the task of table text extraction can be decomposed into three stages. The first stage is to detect tables in the document or datasheet full page images and remove images without tables from being processed in the following steps. The second stage is to detect the structure of the table. We suggest decomposing the structure of the

table into rows and columns in order to capture any hierarchical layouts that may appear in the table. The final stage is to extract the text from each cell / row / column. Methods utilized in the first step must be robust enough to tackle the variations in layouts and styles of datasheet images and the tables within them. To avoid processing images without tables in the later stages of the processes it must also limit the number of images without tables.

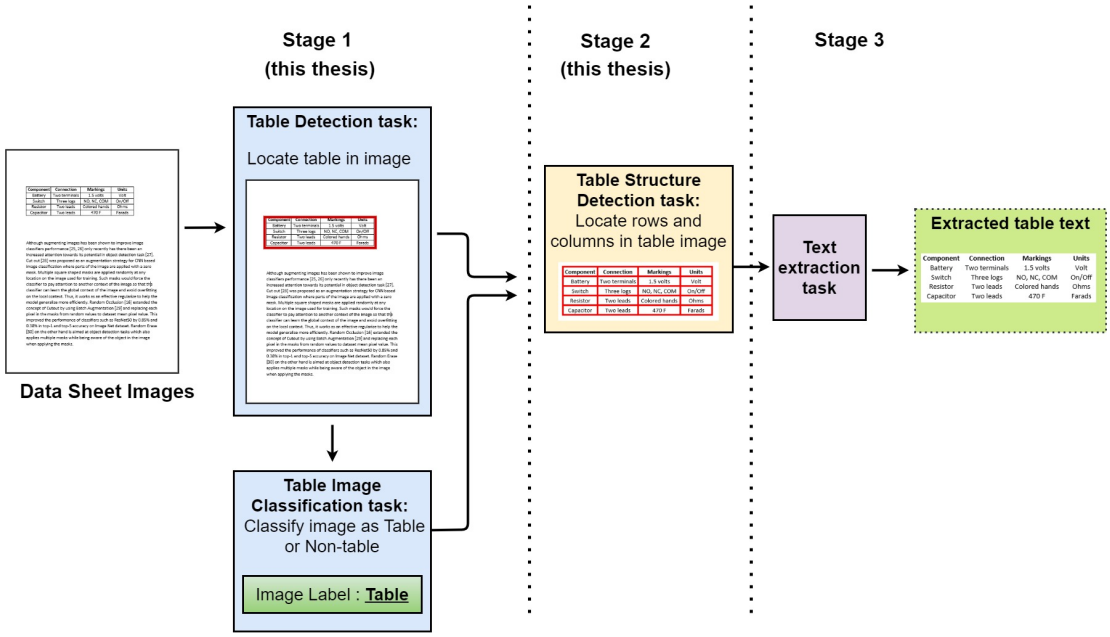


Figure 1.1: **Table text extraction pipeline. This thesis deals with stage 1 and stage 2 of the proposed pipeline**

As shown in Fig. 1.1 we propose addressing table detection and table image classification in the first stage of the pipeline. The major challenge with developing a general solution to address these two tasks is the ability of the solution to adapt to the variations in table structures and to be able to distinguish tables, text and figure objects in the image. As shown in Fig. 1.1 we propose utilizing the detected table region as an input image in the second stage of the pipeline to address the task of table structure detection. Similar to the case of the first stage, a generalized table structure detection solution would have to handle high intra class variance that is caused by the different table structures that appear in such images. This high intra class variance is brought on by the presence or absence of merged cells and presence or absence of line separators between table content.

Previous works for table detection [12, 13, 14, 15] and table structure detection [16, 17,

14] have utilized deep learning based approaches to address these two task. Furthermore, deep learning based solutions have proven to have great generalization potential. Hence, we propose addressing the tasks of table detection, table image classification and table structure detection in images with a deep learning based approach.

1.1 Motivation

Document or datasheet images contain tables with important information such as product specifications. This information is laid out in the most concise format that can be easily interpreted by members of organizations that are familiar with the layout of information. As supply chain organizations such as manufacturers expand the range of products the types of datasheets also continue to evolve. In order to analyze the table text present in these datasheets it needs to converted into a format that can be queried upon so that comprehensive analysis can be conducted before taking critical decisions. Thus a general solution is needed that can detect such tables, extract the text and make the text available for analysis. However the major challenge for developing a solution to identify tables in datasheet images is the infinite number of structures that the tables can take up. In addition, the solution would also have to distinguish between tables, figure and text surrounding the table.

The variations in the structures of tables is brought on by the presence or absence of merged cells in the rows and columns of the table. Tables with merged cells denote a hierarchical layout of information while tables without merged cells denote a structured / regular layout of data with the rows and columns neatly aligned with each other. In addition tables can also be present with line separators between the cells or they may not contain any line separators. In some instances the tables may have only some rows and columns with line separators. Furthermore, tables may also have a border around them separating the table text from the surrounding text or it may not have any border around it. A general table identification and structure recognition solution would have to consider the infinite possibilities of such tables in order to capture all tables and recognize the layout of information in each table.

State-of-the-art solutions for tasks such as image classification and object detection utilize deep learning-based solutions due to the generalization ability of the same architecture to detect different types of objects such as natural objects [11] or objects in datasheet images [14]. Thus, we formulate the tasks of table detection and table structure detection as an object detection task. We suggest using the document or datasheet pages as an image which serves as an input to a deep learning-based solution. Here the table and it's internal

components such as rows and columns can be treated as objects to be detected by the proposed deep learning solution. We put forward one deep learning solution to address the task of table detection and table image classification and another deep learning solution to address table structure detection.

1.2 Research Objectives

The intent of this work is to build a solution that can take a datasheet (document) page image as an input and identify tables, recognize their structures and extract the table text. Thus, in this work we propose a table text extraction pipeline with three stages. As shown in Fig 1.1 the first stage is table detection, where the input is a full page image. This image can be a PDF datasheet or document page that is converted to an image or a scanned image. At this stage a deep learning-based solution is tasked to accurately detect and provide the locations of each table in the images. The objective at this stage is to develop a robust and generalizable table detection solution that can accurately provide the location of any table in a document from any domain.

Furthermore, as shown in Fig 1.1 we propose addressing the table image classification task by utilizing the detection results of the table detection solution at this stage as well. Along with the table detection results, the objective here is to classify the images into two classes; images with tables and images without tables. As a result, the images with tables and the locations of detected tables can be sent to the next stage thus eliminating the need to unnecessarily process images that do not have tables. Thus, the cumulative objective of this work is to develop a single solution that addresses both table detection and table image classification in a single inference.

The second stage addresses table structure detection, where the detected tables from the first stage are cropped and used as an input image. The objective here is to develop a deep learning-based solution to detect the rows and columns of the table while also considering the hierarchical layout of data. The layout is considered as hierarchical if there are merged cells in the table. The final stage is table text extraction where the text from the detected rows and columns can be extracted and used for further analysis. In this work we propose solutions towards solving the first two stages of the table text extraction pipeline. As shown in Fig 1.1 in the first stage we solve the table detection and table image classification task and in the second stage we solve the table structure detection task. The objective is to provide solutions that can handle the variations in layouts presented by tables in datasheet (document) images.

1.3 Contribution

On analyzing the previous work conducted on document analysis towards table text extraction we proposed a pipeline with three stages as shown in Fig 1.1. In this thesis we provide details on steps taken towards building a generalized solution for the first two stages of the pipeline. We propose utilizing one deep learning-based solution to solve the table detection and table image classification task. Furthermore we suggest utilizing an additional deep learning-based solution to solve the table structure detection task.

Our main contributions can be encapsulated as follows:

1. In this work we propose TableDet, a deep learning-based solution towards solving the table detection task in datasheet (document) images. TableDet is built with a Cascade R-CNN [11] architecture and a deformable ResNet backbone. In addition it also utilizes Complete IOU loss [18] to enhance its detection performance.

2. Furthermore we propose Table Aware Cutout augmentation for datasheet (document) images with tables to help deep learning-based models such as TableDet to improve their table detection performance. We also propose a dual step transfer learning process which utilizes Table Aware Cutout augmented images to train the TableDet model. We show that the model trains quickly and provides highly accurate results when it is trained with out two additional proposals.

3. We introduce the table image classification task where the images that have tables are identified and kept for further processing which the images without tables are not further analyzed. We display the impact of using the improved table detection results of the TableDet model towards the task of table image classification. We show that the results of TableDet can be utilized to accurately get all full page datasheet (document) images with tables without any additional training.

4. This work puts forward TableStrDet, a deep learning based model to solve the table structure detection task using the table detected from stage 1 as an input. The TableStrDet model comprises of two Cascade R-CNN [11] architectures each equipped with deformable backbones and Complete IOU loss to improve their detection performance. One architecture is used to detect and classify rows into two classes while the other architecture is used to detect and classify columns into two classes working in parallel. We show the impact of detecting four classes of objects towards accurately capturing table content with hierarchical layout irrespective of the presence of line separators in the table.

1.4 Thesis Outline

This thesis comprises of seven chapters where we describe previous works and our own contributions towards solving table detection, table image classification and table structure detection in datasheet (document) images. In Chapter 2 we provide details about the previous works have been conducted to address the table detection and table structure detection tasks. We describe in brief the work that utilized rule based methods in Section 2.2 before the feature extraction impact of Artificial Intelligence based solutions were witnessed. In Section 2.3 we describe the Machine Learning based solutions that address these two tasks and in Section 2.4 we described the latest works that utilize artificial neural networks or as it is formerly known as deep learning based solutions to address these two tasks.

In this thesis we address table detection and table structure detection as an object detection task on images. In Chapter 3 we describe the R-CNN based architecture to detect objects in images and the augmentation strategies that are utilized to address these tasks. We provide details on the core components of the R-CNN based object detection architecture utilized in this work in Section 3.1. The image augmentation strategy that we developed to enhance the table detection capability of the object detection model is described in Section 3.2.

In Chapter 4 we describe our proposed table detection solution. We describe the steps taken to develop the presented table detection solution in Section 4.1. We also provide details on the training and model enhancements that enable the detection solution to improve its table detection performance in Section 4.1. We describe the training and test datasets for this task in Section 4.2 and the training process in Section 4.3. We display the performance improvements that our proposed approach achieves in Section 4.4 and conclude the chapter in Section 4.5.

In Chapter 5 we describe our proposed solution to address table image classification in datasheet (document) images. We utilize the table detection results from our table detection solution to address this task and hence we describe how we generate the image labels in Section 5.1. We describe the public and closed test sets that we use to evaluate the proposed solution in Section 5.2. In the Section 5.4 we present our analysis that led to the development of the proposed solution. Finally we summarize and conclude the chapter in Section 5.5.

In Chapter 6 we describe our proposed table structure detection solution in detail. The steps taken to develop the presented solution is provided in Section 6.1. In Section 6.2 we describe the datasets used for evaluation and list out the training details in Section 6.3.

In Section 6.4 we show the analysis we conducted to develop the proposed solution and provide a brief conclusion to the chapter in Section 6.5.

Ultimately we conclude this thesis in Chapter 7 and provide a brief summary of the performance achievements that each solution has provided. In Section 7.1 of Chapter 7 we lists out the potential future agenda for the work developed in this thesis.

Chapter 2

Literature Survey and Background

Governments and organizations around the globe store, share and present information in paper based or electronic documents. These document display the information through various document components such as figures, tables or paragraphs of text. The important information in these documents is generally presented in tabular format so that readers can receive information in an articulate format without having to read large paragraphs of text. By placing the content in a tabular format, the critical information that the author of the document tries to convey can be easily summarized to save the comprehension time of the reader. Furthermore, by presenting the information in a tabular format the important details can be easily identified by the reader and can help the reader draw useful conclusions efficiently.

With the advent of Industry 4.0 and the pressing need for automation [2] most organizations and Governments rely on electronic documents to function efficiently. The most common format of electronic documents are either PDF or online (HTML) format [19]. The PDF documents could contain digitally born document pages or scanned images converted into pages. Identifying the table in such PDF documents and making the text from the tables available for analysis can help organizations take a step towards automation. Thus, the previous research towards extracting such tabular data has been focused on addressing this task in two phases; table detection and table structure detection [20].

Zannibi et al. [20] propose that generalized solutions that addresses both table detection and table structure detection tasks would have to consider the physical and logical structures [20] of such tables to perform both tasks efficiently. However table content are presented with various layouts. In some instances tables appear with line separated rows and columns [14]. In other instances the content in the table can appear in a hierarchical

layout [21] to display the relation between common content in the table.

Thus, in this section we provide an overview of previous works that have been conducted for both tasks. The works are divided into three categories based on the approach taken to solve the tasks; Rule based, Machine Learning based and Deep Learning based. We provide a brief description of each approach in Section 2.1. We describe the Rule based solutions in Section 2.2 and the Machine Learning based solutions in Section 2.3. In Section 2.4 we describe the latest Deep Learning based solutions to address these tasks.

2.1 Introduction

Several works have been conducted towards addressing table detection and table structure detection in multiple formats of digital documents. In this chapter we differentiate these approaches based on the type of solution that was develop to address these tasks. We have categorized the approaches into three types. The first is rule based approaches where we describe how image filters and heuristics that were previously used to address these tasks. The second is Machine Learning based approaches which involves utilizing statistical based and non artificial neural network based approaches such as Logistic Regression to address these tasks. Lastly we describe Deep Learning based approaches which involve the latest table detection and table structure detection solutions that utilize artificial neural networks to segment full page images and extract tabular data.

2.1.1 Rule based approaches

Before the emergence of Artificial Intelligence based approaches previous works for document analysis relied on heuristics and image filters to extract tabular data. The rules defined by previous approaches were not generalizable to documents from all domains of the industry. However, they were well defined and robust enough to work with documents in specific domains such as claim documents [22] or business letters [23]. Some approaches were developed for specific use cases such as annotating the table of contents in a document [24].

Furthermore, some methods were developed to build query databases for users with tabular text extracted from HTML based documents [25]. On the other hand, some previous PDF based solutions chose not to convert the pages into images but instead utilized techniques such as identifying important text such as "Table" or "TABLE" [26] or identifying graphical ruling lines and column style white spaces [27] to identify the potential

locations of tables in the document. Other PDF based methods utilized the PDF meta data [28] to facilitate the detection of tables in PDF documents.

In addition to detecting tabular data in PDF and web documents, previous works have also utilized document images before Convolution Neural Networks [29] were applied to address tabular detection. These solutions developed pixel maps to identify blocks of text [30]. These maps were developed with either binary filters [31] or morphological filters [32]. Based on these maps the image could be divided into components which were then labeled as table or text with additional heuristics [32]. To further improve their tabular detection accuracy solutions also used line detection to detect intersecting lines of the table in the image [33].

Furthermore, certain techniques could split the document image into multiple sections based on lines and spaces and developing a bottom up approach to combine the parallel lines to determine if the text was a table or not [34]. Some techniques resorted to character recognition where a tree data structure would be developed to denote the components of a document page starting from the bottom with words as the leaf grouping together to form a table [35].

2.1.2 Machine Learning based approaches

Machine Learning based approaches rely on statistical based algorithms such as Logistic Regression which require the features to be clearly defined in order for the algorithm to function efficiently. On the other hand, Deep Learning based approaches utilize artificial neural networks which have automatic feature extraction capability. Thus, we segregated the two approaches in this work to emphasize on how the previous works extracted features from images and PDF's to develop tabular data extraction solutions.

The early works for table detection and table structure detection resorted to segmenting the image into multiple components to comprehend the possible layouts of the document irrespective of whether a PDF or image was used. Some techniques utilized Machine Learning algorithms such as binary decision tree to classify the components of the image as table or non-table [36]. With the emergence of Support Vector Machines (SVM) [37] some solutions began utilizing SVM's to address these tasks.

SVM's were successfully utilized for HTML documents [38] due to the availability of HTML tags which were used to generate features. They were also utilized for PDF documents where the features of each page were already extracted by off the shelf PDF text extraction softwares [39]. On document images as well SVM's were able to identify components of a document image with a higher success rate than other Machine Learning

algorithms. The only drawback of utilizing SVM's for document images was its reliance on accurate features from the images. Certain solutions applied multiple image filters to make the features of a table more prominent [40, 41]. These features were then fed to an SVM so that it could classify the page components accurately.

As the number of Machine Learning algorithms continued to grow certain approaches applied an ensemble of such algorithms to improve their tabular information extraction accuracy [42]. The unique aspect among the previous Machine Learning approaches was the intuitive techniques that were applied to generate features for each Machine Learning algorithm in advance and irrespective of the format of the digital document.

2.1.3 Deep Learning based approaches

The world of Computer Vision began to witness the impact of Convolution Neural Networks with the introduction of AlexNet [29] which could address image classification on large images without any manual feature extraction. This further led to the development of multiple large scale image classification architectures such as VGG [43] and ResNet [44]. These architectures not only proved useful towards addressing image classification but also towards other image processing tasks such as object detection and image segmentation by acting as automatic feature extractors. Thus object detection architectures such as Faster R-CNN [45] and YOLO [46] were developed to address the tasks of detecting natural objects with a higher accuracy.

Furthermore, these deep learning object detection architectures also began to display their generalization capability as multiple works in domains other than natural images began to utilize these architectures [47, 48]. One such domain is document analysis which continues to utilize these architectures to improve the quality of tasks such as table detection and table structure detection in document images. In addition, the enhancements in deep learning have also enabled works for PDF based documents which have the additional benefit of utilizing PDF meta data to segment the PDF pages into multiple page components [49].

For document images, most deep learning approaches utilize one deep learning architecture to address table detection and another architecture to address table structure detection [50, 51]. Thus, most previous works solve only one task at a time. They utilize state-of-the-art object detection architectures such as Faster R-CNN [45] and Mask R-CNN [52] to detect either tables or rows or columns of tables. Previous works for table detection have enhanced such deep learning architectures to address not only table detection [12, 13]

but also table structure recognition [17, 53, 16]. Furthermore, Deep Learning feature extractors such as VGG [43], ResNet [44] and ResNext [54] can also provide rich features to these architectures. Thus, previous works were able to enhance the quality of their table detection and table structure detection results by utilizing the features provided by these feature extractors.

In addition to that deformable convolutions [55] also provide more features that would ordinarily be ignored by standard convolutions such as multiple scales of tables. Thus previous works have also applied deformable convolutions [55] to further improve the performance of their presented solutions [56]. Other techniques have resorted to adding more features about the table such as the corners of the table regions to boost the table detection performance of their proposed approach [57].

While deep learning architectures facilitate automatic feature extraction, some tabular data extraction works have enhanced their solutions by providing additional features to their table detection solution to enhance the quality of their presented solution. For instance utilizing CNN and the component interdependence capability of CRF to segment a full page document image and to identify clusters of similar components [58]. Some techniques have utilized the feature extraction capability of CNN's to develop saliency maps which improves the quality of segmenting document images and helps the table detection solution to identify tables more accurately [59].

Some techniques have also considered the possibility that table appear without line separators between rows and columns [60, 14]. Thus, to improve the table detection performance of their approaches they have generated additional features with the help of Generative Adversarial Networks [61]. In addition to the features provided by CNN architectures, solutions have also extracted document specific features to improve the table detection and table structure detection capability of their solutions [15, 62].

2.2 Rule based approaches

Early works such as the system presented by Nagy [22] for automated text extraction heavily depended on the type of document that was to be processed. Depending on the type of document a different method of text extraction would have to be applied. For instance, if the document was a simple claim document for a patent, certain sections of the text would be more important than the rest. Thus a system with knowledge about the features of such a document could easily detect and extract the important text from the document. However, unstructured documents such as technical journals papers would

require a interactive system. In this system the human operator would use a joy stick to locate the important text and then with the help of several filters would be able to highlight, extract and store the required text. Since, the important text in the document was found in the cells of tables the presented system focused directly on the detecting and extracting tabular text. Thus, the system did not specifically focus on detecting tables in the images.

Following the document analysis work proposed by Nagy [22], Wahl et al. [30] put forward a block segmentation technique to analyze digital documents. The authors proposed that a page from the digital document can help generate pixel bit maps of the size of the image which contained pixels of white ($0's$) and black ($1's$). These maps indicated the location of page objects such as table, text, figures, horizontal and vertical lines. The $0's$ denoted the white spaces while the $1's$ denoted the blocks of page objects that were adjacent to each other. The presented system generated horizontal and vertical bit maps to provide context about the locations of each page object. It also added a horizontal smoothing bit map to capture any broken text lines that were not captured by the other bit maps. Each pixel was then labeled from left to right based on the adjacency of the black pixels. Utilizing the features of the blocks such as height and ratio of block pixels to bounding lines each block was classified as one of the five page objects.

Table Information based Text Inquery (TINTIN) was put forward by Pyreddy et al. [63] to detect tables in digital documents by using a set of heuristics. The authors recommended that the task of extracting the table text and making it available for queries could be addressed in four steps; table extraction, table component tagging, indexing, retrieval. Table extraction utilized a Character Histogram Graph (CAG) data structure. This structure develops a histogram indicating contiguous blocks of text and can identify a table based on the gaps between columns in a table. The component tagger system then labels the components of the table such as column header, table text and captions using a set of heuristics. Gap Structure Heuristic captures the horizontal gaps in the tables and Alignment Heuristic captures the layout of gaps and text in multiple lines to denote if the content is within the table. The captions and table text lines would appear in a pattern which would be captured by the Pattern Regularity Heuristic. Furthermore, Differential Column and Gap heuristics could differentiate between captions (which appear as text) and table text lines (which appear as numbers). In addition to this, the text were indexed and made available for analysis through queries.

To convert paper based documents into an electronic version for storage and analysis, Jain et al. [31] put forward an image processing technique that decomposes the digital page into multiple components. The authors developed a top-down model that segments the digital page. For this study they utilized technical journal papers which have a uniform

layout and scanned the pages to use as input images. A binary filter was applied on the image and then the text and non-text parts of the image were processed separately as a page segmentation step. The small connected components of the binary image were considered as the text parts while the large connected components such as table, halftone image, and drawing were considered as the non-text parts. The non-text parts were processed for region identification based on a series of heuristic to determine if the region was a table, drawing or text.

Kieninger et al. [35] developed the T-Recs system which utilizes character recognition to solve table structure detection in documents. The authors recommended dividing document into blocks of words, where words are considered as the elementary component of the document instead of lines. Each word is denoted with a bounding box to represent its location in the document. If the neighbouring bounding boxes appear close to each other and if they form a pattern that shows a vertical (for columns) or horizontal (for rows) alignment of text then these bounding boxes which combine to form a block are considered as part of a table. To reduce any errors in labelling such blocks as a table, the presented approach took steps to combine such blocks into table tiles. Each table tile was a cell created by intersecting lines of potential row and column blocks. The proposed system was further enhanced by Kieninger et al. [23] to improve its table detection performance specifically for business letters. This new system was called T-Recs++.

TabFinder was a unique document image processing technique proposed by Cesarini et al. [34]. The process starts by dividing the document page image into a data structure called the Modified X-Y (MXY) tree [64]. The tree would represent all components of the document image where the leaves were the lowest components of the image after splitting it based on white spaces and lines. Then based on the presence of parallel lines in the image, the system would decide if the text block within those lines was a table or not after climbing up the MXY tree. Furthermore, this process was able to outperform the OCR systems available at that time on two public datasets for the task of table detection in document images.

On the other hand, Gatos et al. [33] proposed a series of image analysis techniques to detect tables in any type of document without any training or first hand document based heuristic knowledge. The presented approach consisted of three steps. First, image processing, where a binary filter was passed over the image to generate a binary image. In this step, the alignment of the page in the image was also corrected by using skew filters. Second, horizontal and vertical lines were detected throughout the image. Only those lines that had intersections would be grouped together to be considered as a table. Lastly, table detection and reconstruction where the intersecting lines from the second step would be used to identify the cells locations of the table. However, one drawback of this system was

the dependence of the system to have tables with line separators between the cells.

Instead of utilizing a heuristic based approach, Lin et. al [24] proposed utilizing a statistical based approach to comprehend the table of contents in a multi-page digital document. They used statistical natural language processing and layout analysis to perform parts of speech tagging and then to text mine the entire document. They also utilized OCR systems to mine the text and to avoid any errors from either text mining processes. The proposed system was aimed at identifying the table of contents of the document. Furthermore, with data retrieved from text mining the system was able to produce links from the table of contents to each chapter or article within that document. The accuracy of these links was enhanced by matching the text and page number in the table of contents to the chapter / article tile at that page number. A benefit of utilizing this approach is that this solution requires no additional training to identify table of contents in documents. However, this application of this process is only limited to detecting table of contents.

Web pages also provide tables with important information that can be essential for analysis. One possible method of extracting the content in this table was proposed by Gatterbauer et al. [65]. The method was unconventional since it avoided using the page as an HTML page but rather utilized 2-D topology and typography. On the other hand, Pivk et al [25] proposed TARTAR (Transforming Arbitrary Tables into Frames) method to extract table content in HTML documents using HTML tags to make the data available for query answering. The presented system utilized the tags in the document to understand the layout of the table. Although the authors claim that the method can be utilized on tables in any document format, the process is heavily reliant on the presence of tags to indicate the location of tables and the text within them. Rather than focusing on detecting the tables the method is enhanced for developing an inference engine that can serve as a query answering system to users. The proposed method can accurately capture multiple layout types of HTML tables and can provide the data in these tables in an F-logic format that helps the inference engine to quickly answer any queries that a user may raise.

Lui et al. [26] proposed an approach to detect tables in PDF files directly instead of converting the files into HTML documents or pages into images. The proposed approach addresses the task of table detection by first solving text sequence errors that are faced by PDF text extraction tools. The authors propose two algorithms to recover the text sequences. One considers the document column information and the other does not consider the document column information. Furthermore, the proposed method also detects the keywords in the document such as "Table, TABLE, Form, FORM," so that it can identify the table boundary. The authors determined that the second algorithm detected tables more accurately than the first algorithm on both single and cross column tables.

PDF-TREX proposed by Oro et al. [66] was a heuristic based bottom up approach that recognized table structures in PDF documents using spatial features. On the other hand, Fang et al. [27] proposed a method that utilizes content layout analysis and visual features to address table detection in PDF files. The process first separates the content of the page based on graphical ruling lines and white space delimiters. The aggregated text blocks are then inspected for visual separators such as graphical lines and white spaces to denote the presence of a column in a table. If a series of columns are detected then the aggregated text block is denoted as a table. Unlike pdf2table [67] and PDF-TREX [66] solutions presented by Fang et al. [27] is applicable on single and double-column pages.

Furthermore, the earlier works for table detection and table structure recognition in PDF documents did not stop at simply identifying the table and cell locations. Liu et al. [28] proposed TableSeer as an additional approach that crawled PDF documents on the web, identified table regions, indexed the content of the tables and provided a system for query answering. Tables were detected by labeling and merging sparse lines. However, TableSeer was highly dependent on the presence of table captions and meta data without which it would not recognize a table. On the other hand, TabbyPDf was put forward by Shigarov et al. [7] as a heuristic based approach to detect tables in untagged PDF documents. Thus, without the use of meta data from the documents the authors developed a web-based system that identified and recognized table structures to provide a tagged table content in HTML or JSON format for query answering.

Instead of applying a heuristic based approach to detect tables in images Tran et al. [32] applied a morphological approach. The authors proposed a series of steps to classify components of a document page image as a table. First the connected components of binary document image are obtained after applying a morphological image filter. Then based on three conditions the connected components of the image are labeled as table or non-table. The conditions are: existence of table boundaries, presence of text blocks and finally a check to see if the text is aligned vertically. The authors performed these experiments on the ICDAR 2013 [68] test set and is one of the first works to address table detection in images from this test set.

2.3 Machine Learning based approaches

Machine Learning based solutions for table detection address the problem as an object recognition task. Wang et al. [36] proposed classifying each zone (region) in the document image as an equation, table, drawing / map, ruling line, logo or multiple other page object types. To classify each zone the authors utilized a binary decision tree [69] and pruned

the tree to prevent it from overfitting. To further enhance the zone classification the presented approach applied a contextual constraint as a Hidden Markov Model (HMM) [70, 71]. Following that the Viterbi algorithm [72] was applied to identify an optimal classification result. This constraint was applied on only a small subset of zones that the system found difficult to classify during its training phase. In their subsequent work [38] the authors applied decision trees and Support Vector Machines [37] to web based documents to classify blocks of text as genuine tables or non-genuine tables. This work displayed the impact of using SVM with an RBF kernel for this task. The presented solution achieved the highest F1-score in accurately detecting genuine tables in HTML documents.

Shetty et al. [73] put forward a machine learning approach to segment scanned document images. The authors propose classifying the sections of the images into one of three classes; hand writing, noise and machine print. Tables are not directly detected by this approach. However, the use of Conditional Random Fields [74, 75] to segment the images provides an interesting potential. This interest is brought on due to its ability to capture the inter-dependence of the three object classes in scanned document images more accurately than other techniques.

Unlike other techniques that utilized images or web based documents, Ng et al. [40] utilized plain text Wall Street journal news documents to show that tables can be detected by machine learning systems in other types of document formats as well. The authors utilized a C4.5 model and trained it to detect occurrences of lines in the text. The system first detected horizontal and vertical lines to identify the location of the table. Next, it identified vertical lines in the detected location to identify columns and horizontal lines to indicate the location of rows. The authors propose a similar process of table information extraction as our work in this thesis where the table detection is followed by table structure detection. A similarity of this technique with our work and other image based table recognition solutions is the fact that these techniques require additional annotations to denote the location of the table and its components; rows and columns.

Hidden Markov Model (HMM) [70, 71] is a probabilistic graphical model that can develop a high quality understanding the interdependence between objects. Thus, Silva [76] proposed a machine learning method where classifiers would provide features to help develop a probability graph with HMM in order to understand the content layout of a document page. The author utilized text documents converted from financial PDF documents in this work. This was one of the first works to show that probability graph methods such as HMM can be utilized to accurately locate tables in text.

Kasar et al. [41] utilized the row and column line separators to detect tables in document page images. The authors used a series of image processing filters (Gaussian, Black-

hat and Adaptive threshold) to make the lines in a document page more prominent. 26 low-level features were then extracted from groups of intersecting lines that were displayed in the final filter output and provided to an SVM model with a RBF kernel. This work was evaluated on document images from the corpus of Arabic, English and French documents. A limitation of this work is its dependence on the availability of line separated rows and columns within a table which is not always the case for all tables in document images. Tables also appear without lines and in some instances only lines to indicate the start and end of the table content.

Distance Supervision [77] was a heuristic based learning process to develop weakly labeled annotations for PDF documents. Fan et al [42] utilized this process to develop annotations in PDF documents from ACL anthology and ICDAR 2013 dataset [68] to address the task of table detection and structure recognition. The authors propose pdfExtra as a system to address these tasks. It utilizes off the shelf software such as Apache PDFBox and OpenNLP to detect and recognize words that exists on the same line and to provide vertical and horizontal co-ordinates for each line. It utilizes the weak labels from distance supervision process output along with the outputs provided by the two softwares to generate features for three machine learning algorithms; Naive Bayes, Logistic Regression and SVM. The three classifiers are used as a voting ensemble to decide if each input text line belongs to a table or not.

Tkaczyk et al [39] proposed CERMINE a heuristic and machine learning based system to segment and extract meta data of PDF documents. Although CERMINE did not focus only on detecting tables in the document, it did identify tables along with other components within a document page. The system first understands the basic layout of the page by utilizing a combination of heuristic and off the shelf software to segment the page into multiple zones. The zones are then classified by a SVM into four broad categories; meta data, references, body and other. The zones are further classified based on the meta data detected in the previous step. At this stage the tables are accurately detected by another SVM.

2.4 Deep Learning based approaches

Hao et al. [49] proposed a Convolutional Neural Network (CNN) based approach to detecting tables in PDF documents. The proposed system first identifies potential table regions in each page. These regions were identified based on the presence of horizontal and vertical lines to create a grid or groups of text that appeared in a vertical alignment to represent a column. Thus, this step not only captured table regions but also non-table

regions such as charts that appeared as grids. The CNN was then trained on images of these regions to classify the regions as table or non-table. To improve it's classification performance, features from the PDF meta data such as font and color were also provided to the CNN. This system was tested on the ICDAR 2013 [68] test set.

DeepDeSRT was proposed by Schreiber et al [50] which was a deep learning based solution that only used images unlike the solution proposed by Hao et al [49] which used images and PDF meta data together. DeepDeSRT system consists of one deep learning system to detect tables in images and another deep learning system to detect rows and columns in the detected table which is used as an input image. The authors propose utilizing Faster-RCNN [45] a two stage object detection architecture with a VGG-16 [43] backbone to detect tables. The authors utilized a FCN based segmentation model [78] to solve the task of detecting of rows and columns. The FCN system address this task as a semantic segmentation task where each pixel was labeled as a row or column.

The ICDAR 2017 POD competition [79] provided 2417 document images for competitors to detect and classify each page object. Li et al [58] used this dataset to classify objects as tables, figures, text and formulas. The system proposed by this work first segmented the images into column regions and then line regions. Then each region was clustered into one of the four page object classes by a CNN backed CRF architecture. In the following step the neighbouring regions belonging to the same cluster would be labeled as an object from one of the four classes.

Kavasidis et al. [59] present a unique approach to detecting tables that utilizes the salient features of a table to help the presented system distinguish tables from other page object types. The authors propose generating saliency maps for four object types; pie charts, bar charts, tables and line charts. The maps are generated by VGG-16 convolutional network where the penultimate layers of the network utilize dilated convolutions [80] to help it generate more accurate saliency maps. Individual binary classifiers for each class are trained in this system to detect these objects of these four classes. Individual CRF [74] are also utilized to help the construction of these saliency maps. Thus, for each object type a saliency map is generated by the system in order for it to distinguish between other object types.

Traquair et al [12] utilized a Faster R-CNN [45] architecture with a VGG-16 [43] backbone to develop a table detection solution for document images. The presented architecture could out perform the table detection performances achieved by DeepDeSrt [50] and Kavasidis et al. [59] on the ICDAR 2013 test set. To address the table structure recognition task in images of detected table regions Kara et al. [53] presented a semantic segmentation approach to solve this task. The authors utilized Mask R-CNN [52] architecture with

ResNet-50 [44] backbone to segment the detected table image into rows and columns.

In addition to detecting tables [12] and segmenting the table into rows and columns [53], Kara et al. [17] also purposed an end-to-end system to extract tabular information from images. The authors proposed detection of intersecting lines in a agreement system along with the object detection outputs of Mask R-CNN to detect tables and segment the tables into rows and columns. Utilizing both types of outputs the authors show that a holistic system can accurately address both table detection and table structure detection in datasheet (document) images.

Jiang et al. [13] also utilized a Faster R-CNN architecture to address the task of table detection in images. In addition to that, the authors improved the detection performance of the architecture by utilizing a ResNet-50 [44] backbone. Furthermore, the authors also proposed utilizing Kullback-Leiberg loss and Variance Voting [81] instead of the standard $l1$ bounding box regression loss to refine the quality of the detections. On the task of table structure recognition Jiang et al [16] proposed detecting cells instead of rows and columns in the image of the detected table region. The authors proposed TabCellNet architecture to solve this task. The architecture comprises of Hybrid Task Cascade [82] architecture which receives features from a CBNet backbone [83]. The CBNet architecture employs dual ResNext101 [54] architectures to provide rich features. Furthermore, the authors also present the impact of utilizing Soft-NMS [84] instead of the standard NMS.

Siddiqui et al. [85] were the first to propose utilizing deformable convolutions [55] in place of standard convolutions to capture the multi-scale and multiple orientations of tables in document images. The authors proposed DeCNT architecture to address this task. The architecture comprised of a Faster R-CNN [45] architecture with a deformable ResNet-101 backbone to provide rich features. The authors also present the impact of employing a deformable ROI pooling layer in the DeCNT architecture to improve its table detection performance. The deformable convolutions capture the variations in scale and orientations of tables with higher efficiency that standard convolutions. Thus, the DeCNT method [85] was able to achieve the highest F1 score public table detection test sets, including the complete ICDAR 2013 [68] test set.

Siddiqui et al. [56] also employed the adaptive nature of deformable convolutions to develop a another solution for table structure detection in the images of the detected tables. The authors proposed DeepTabStR architecture which comprised of a two separate deformable FPN [86] each with a deformable ResNet-101 [44] backbone to solve this task. This was one of the first works to display the impact of using a two separate architectures to detect rows and columns in a single inference. This work presents the significant improvements brought on when one architecture detects rows and another architecture

detects columns in the images in parallel rather than using a single architecture to detect both types of objects.

Sun et al. [57] were the first to propose the detection of table corners in document images to precisely detect the table location in the image. The authors propose utilizing a Faster R-CNN [45] architecture with VGG-16 [43] backbone to detect tables. Earlier works such as CornerNet [87] which considered the corners of natural objects displayed an improvement in object detection performance for naturally occurring objects. Thus, Sun et al. [57] proposed detecting the corners of the table as well as the table region within each image. The proposed system provides the locations of the table and each of its four corners. The corners are then refined to improve the overall table detection performance of the presented system.

Table in document images can appear with or without line separators between the rows and columns of the table. Tables with such line separators can be easily distinguished from surrounding text objects in the images. However, when the tables do no have such ruling lines a table detection solution must rely on observing the alignment of text in potential table region to accurately detect the table. Thus, Li et al. [60] proposed an additional feature generator to provide features about such tables to an object detection model to improve its table detection performance. The authors use a generator and discriminator architecture to develop this feature generator inspired by the architecture of a Generative Adversarial Network (GAN) [61]. It is trained to distinguish between real (tables with ruling lines) and fake (tables with ruling lines erased) images so that it can learn to identify tables based on their content layout. In this work a Mask R-CNN [52] architecture is used to detect tables in images. Along with the features it receives from its backbone of its standard architecture it also receives rich features from the trained feature generator.

TableNet was proposed by Paliwal et al. [15] to address table detection and table structure detection in two public datasets as a semantic segmentation task. The system is built as an encoder - decoder system. The encoder for TableNet was a VGG-19 [43] convolutional backbone. The authors used two decoders, one for detecting tables and one for detecting columns within the tables. Both decoders were trained with the single encoder so that the encoder could learn to distinguish between table and column objects in the image. Furthermore, the input to this system was not just the image and the annotations for the tables and columns. Instead, the image was pre-processed by an OCR system to identify and color code the common data types in the image. For instance, the text values was color coded with one color while the numerical values were color coded with another color.

This pre-processed image was fed to the TableNet system as a means of providing visual

features as well as semantic features to improve its detection performance. Simple image transforms were applied to the image for row detection without training the system to detect rows. Arif et al. [62] also proposed a system that highlighted sections of the table and surrounding text with similar data types to help a Faster R-CNN model [45] distinguish between foreground (table) and background (white spaces) regions of the image. However, this system could only detect tables unlike TableNet which was able to detect both tables as well as columns and rows in a single inference.

Siddiqui et al. [51] also proposed an encoder - decoder system to semantically segment the images of detected tables into rows and columns. Similar to the segmentation process of Paliwal et al. [15] this process would also classify each pixel to either belong to a row or column or white space. The authors utilized a Inception ResNet v2 [9] model as an encoder to extract features from the images. The decoder consisted of two series of convolutional layers, one for rows and another for columns to classify each pixel as a row / white space or column / white space respectively. The TableNet solution, detected the table along with its columns in parallel. However, for the table rows it utilized certain transforms to detect the rows instead of utilizing the encoder - decoder architecture. On the other hand, the encoder - decoder solution presented by Siddiqui et al. [51] detected the rows along with the columns using the features extracted by the encoder in a single inference.

The text in rows generally appear in a horizontal alignment while the text in columns generally appear in a vertical alignment [15]. Thus, Khan et al. [88] determined that utilizing convolution networks to identify rows and columns in a table image was not efficient. Due to the repetitive nature of rows and columns, Khan et al. [88] proposed identifying rows and columns as a sequential set of objects by utilizing a Recurrent Neural Network (RNN). Furthermore, Khan et al. [88] also compare the efficiency of utilizing Long Short-Term Memory (LSTM) [89] RNN over Gated Recurrent Unit (GRU) [90] RNN and determined that GRU's were a better fit to address table structure detection. Thus, they presented an approach where one Bi Directional GRU [91] would be utilized to identify rows while another Bi Directional GRU [91] would be utilized to identify columns. Similar to previous works such as DeepTabStR [56] and Siddiqui et al. [51] this approach also utilizes two separate deep learning architectures to detect rows and columns in a table image.

Generally many deep learning based solutions for table detection had resorted to utilizing two stage object detection architectures such as Faster R-CNN. However, Huang et al. [92] determined that the advancements in single stage object detection architectures such as YOLO [46] could be tailored and utilized to achieve higher table detection performances as compared to two stage table detection solutions. In order to achieve this, certain enhancements were made to the YOLO deep learning architecture in terms of it's

anchor proposals to facilitate the architecture to focus on detecting tables with a higher accuracy. Furthermore, additional post processing steps were proposed by Huang et al. [92] to help the table detection solution. These steps were empirically determined and included conditions such as the area of the table should have more than 500 pixels and the ratio of width over height or vice versa should be greater than 12.

Prasad et al. [14] proposed CascadeTabNet a single deep learning based approach that detects tables and recognizes the structure of the table in a single inference. The system architecture comprises of a Cascade Mask R-CNN [11] object detection model with HRNet [93] as the backbone feature extractor. The authors propose an iterative transfer learning process to train the system to first detect tables and then classify tables in incremental steps. The systems classifies a table as bordered or borderless depending on the presence or absence of line separators between cells of the table respectively. The system considered tables with partial lines as borderless tables. The authors proposed that the same system could be utilized to segment the table. Bordered tables were segmented into rows and columns by utilizing image based line detection algorithms without any additional training. However, as part of the iterative learning process the system was trained to segment the borderless tables into cells with the help of cell level annotations for such tables. The CascadeTabNet system achieved the top spot at the ICDAR 2019 competition [8] for table structure recognition on modern document images in the ICDAR 2019 test set.

Table 2.1: **Overview of research conducted for table detection and table structure detection in multiple document formats**

Rule based	Machine Learning based	Deep Learning based
[22, 30, 63, 31, 35, 23, 34, 33] [24, 65, 25, 26, 66, 27, 67, 28, 7, 32]	[36, 38, 73, 40, 76, 41, 42, 39]	[49, 50, 58, 59, 12, 53, 17, 13] [16, 85, 56, 57, 60, 15, 62, 51] [88, 92, 14]

Deep learning solutions applied to detect tables and recognize the table components such as rows or columns or cells work purely on the visual features provided in the image. The images in some solutions were highlighted in certain regions to improve the solutions performance but no additional meta data was provided nor available unlike PDF based solutions. All deep learning based solutions utilize the high generalization capability of the available architectures such as Faster R-CNN [45], Mask R-CNN [52], Cascade Mask R-CNN [11] and many more to address table detection and table structure detection in document images. These document images are not limited to scientific reports but also include datasheets. The solutions listed in the aforementioned text have proven to be applicable all types and formats of documents.

Furthermore, Deep Learning based solution rely on large datasets being available to address tasks such as table detection and table structure detection. For such large datasets additional annotations to denote the locations of tables, rows and columns would also be required which is quite an expensive task to develop as it requires a large number of manpower hours to manually annotate these datasets. As an alternative solution that does not rely on Deep Learning could also be utilized to address these tasks. While not as efficient as a Deep Learning solution they can be tailored to work efficiently for a documents that have very similar layouts. OpenCV utilizes Computer Vision rules such as Binary Threshold to identify tables as blocks of aligned text with or without line separators. PDF based solutions such as PDFMiner and OCRmyPDF which use the document directly without converting the pages into images are also alternatives to the state-of-the-art Deep Learning solutions.

Previous works that have address the task of table detection have only detected table and have not detected text and figures. Nor have they used an IOU based loss to improve localization. Nor have these techniques utilized a specialized transfer learning technique such as our proposed dual step transfer learning process. While some works have added some additional augmentation strategies as shown in [14], none have developed a strategy as specific to tables as our proposed Table Aware Cutout. In addition to that, we have utilized additional training examples as compared to many of the previous works. Thus, while our training and model enhancements have vast differences as compared to the previous work done in this task we have ensured that the test sets and the evaluation metrics that were used for this study were exactly the same as those that were utilized by the previous works so that we could fairly compare the impact of our proposed enhancements.

Similarly previous works that have address table structure detection have either detected cells [14, 17] or rows and columns [15, 56]. None of the previous works have classified the rows and columns into regular and irregular classes to capture the hierarchical layout of information in the table. Thus to fairly compare our results with the state-of-the-art solution DeepTabStr [56] we have ensured that the training and evaluation of both our proposed and the state-of-the-art solution are done on the same training and test sets respectively.

Chapter 3

Methodologies

Organizations communicate with each other despite the physical distance between them on a global scale using documents (electronic or paper based). These organizations include manufacturers, component distributors and product designers. They share information through documents such as datasheets which provide component level information. These documents contain valuable information that can help these organizations to effectively deal with their supply and demand requirements. The electronic industry in particular is reliant on accurate and up-to-date datasheets to develop reliable and compatible systems. These datasheets are mostly digitally born documents. They can be PDF based documents or can contain scanned images of paper based documents.

These documents contain valuable information such as product availability, specifications and pricing in tabular format in order to display the data in a concise format. Members of such organization can utilize the data in such tables to analyze and make decisions based on large amounts of structured information [68]. Furthermore, extracting and assimilating the data from the tables for analysis can help organizations to build a sophisticated knowledge management system. This system can support efficient automation as well as evidence and data-driven decision-making [5]. Thus, we hypothesize that the task of extracting this data can be divided into three stages as shown in Fig 1.1. In addition to the pipeline we also propose utilizing images of document pages instead of utilizing PDF documents as inputs to the pipeline.

In this work, we propose addressing table detection and table image classification in the first stage of the pipeline. This stage involves correctly identifying that the image has a table and accurately locating the table in the image. Furthermore, we classify the images as Table (images with a table object) or Non-Table (images without a table object) at this

stage of the pipeline so that we can reduce the number of Non-Table images that would have ordinarily been processed in the next stage of the pipeline if the images are not classified. We present, TableDet, a deep learning based approach that solves both tasks in a single inference. For the second stage of the pipeline we propose that the detected table region from stage 1 is analyzed and the rows and columns of the table structure are detected. For this task of table structure detection we present, TableStrDet, a deep learning approach that decomposes a table structure into four classes of objects; regular rows, irregular rows, regular columns, irregular columns. TableDet and TableStrDet address table detection and table structure detection respectively as an object detection task.

Both models use a R-CNN based architecture to detect objects in images. We describe the core components of the R-CNN architecture in Section 3.1. We also describe the proposed data augmentation strategy for table detection solutions in Section 3.2. Further details about the application, enhancements and performance improvements over state-of-the-art solutions are provided in Chapter 4 and Chapter 5 for TableDet and Chapter 6 for TableStrDet.

3.1 R-CNN based object detection architecture

We address the task of table detection and table structure detection in datasheet (document) images as an image object detection task. We present the TableDet models and TableStrDet model to address these tasks respectively in Chapter 4 and Chapter 6 respectively. Furthermore, the table detection results of TableDet models are utilized to address the table image classification task as well as described in Chapter 5. Both TableDet and TableStrDet models utilize Region Based Convolution Neural Network architectures to address these tasks. In this section we describe the components that were utilized and enhanced to develop these models.

3.1.1 Backbone

Deep learning based feature extractors for images utilize multiple convolution layers in their architectures to extract rich semantic information from images. LeNet [94] was the first to present the benefits of utilizing convolution operations to extract features in place of fully connected neural networks. The functionality of LeNet displayed how convolution layers stacked on top of each other could generate low level features such as shape of the object in the image. In addition to that it did not require any additional

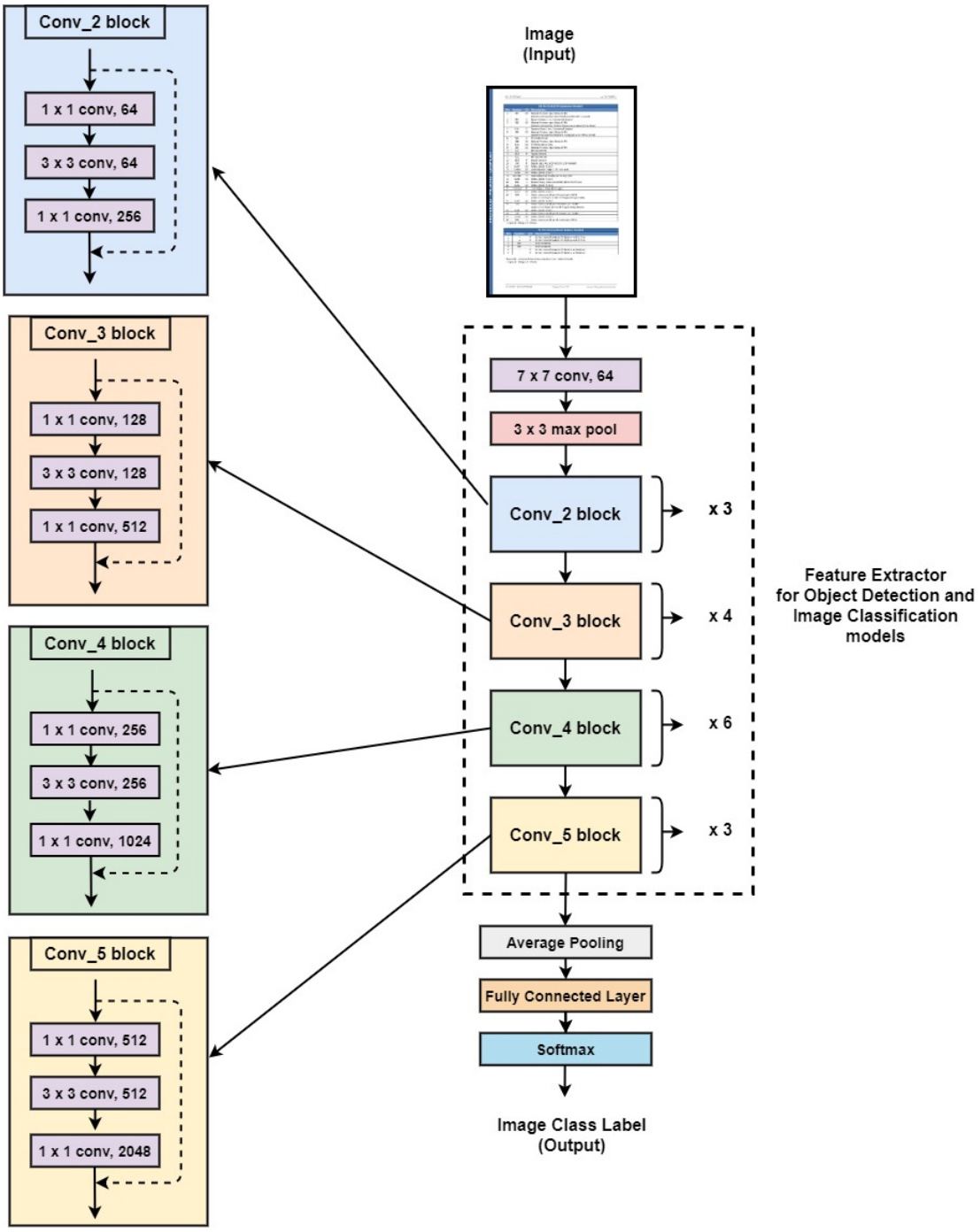


Figure 3.1: **ResNet-50 architecture**

hand crafted rules to be provided in order to generate such image features. In 2012 the AlexNet architecture which was proposed by Krizhevsky et al [29] which enhanced the LeNet architecture by utilizing ReLU [95] activation functions and introducing dropout as a regularization technique. Furthermore, AlexNet contained more number of convolution and pooling layers than LeNet. This deeper architecture provided more finer features along with the low level features that were provided with LeNet architecture. Thus, this developed the hypothesizes that deeper architectures could provide more richer features than shallow architectures / networks.

GoogleLeNet [96] improved on the LeNet network by introducing an inception module. This module utilized multiple small convolution filters in parallel to reduce the total number of parameters in the network while maintaining a deep network with 22 layers. Although it required more memory usage than its predecessors it achieved the highest image classification accuracy on the ImageNet dataset in 2014 [97]. In the same year, Simonyan and Zisserman [43] presented the additional benefits of building deeper models with the VGG architecture. The VGG architecture accommodated 16 to 19 weight layers to provide hierarchical representation of visual features. It achieved the first and second highest performance for the the task of object detection and image classification respectively on the ImageNet dataset in 2014 [97]. Furthermore, it exclusively utilized convolutions with smaller filter sizes ($3x3$) to limit the number of parameters in the architecture while also enhancing its feature extraction potential.

These deep networks are trained with back propagation to improve their output representation. This process involves calculating the gradients to reduce the loss between output generated by the network and the input. Deeper networks have the advantage of extracting multiple scales of features however they face the risk of vanishing gradients due to their large depths. Residual Networks (ResNet) were introduced by He et al. [44] to address this vanishing gradient issue. In addition to that the architecture also address the stagnation in accuracy that deeper networks face when the depth of the network is increased. The ResNet architecture resolves these issues by introducing residual mapping. The mapping process involves an identity addition with the help of skip connections to add the input (x) and output of a small predefined number of convolution layers ($F(x)$). In an ordinary non-residual network the output $F(x)$ from multiple predefined number of convolution layers would be used during back propagation leading to a gradient calculation issue. By performing this identity addition calculation as shown in (3.1) the ResNet architecture not only improves its output representation (Y) as the input is constantly being presented to various sections of the network but also address the vanishing gradient issue.

$$Y = F(x) + x \tag{3.1}$$

Unlike other deep neural networks, ResNet architecture places the convolution layers into blocks of convolutions which we refer to as convolution blocks. Within each convolution block there are multiple convolution layers. As shown in Fig. 3.1 we display four type of convolution blocks each with different colors. He et al. [44] displayed the flexibility of this architecture by proposing that the number of layers in each block can be increased and decreased. Furthermore, the blocks can also be repeated. He et al. [44] present multiple such ResNet architectures with the smallest network only using 18 layers while the largest network uses 152 layers. In this work we utilize ResNet [44] architecture with 50 layers which we refer to as ResNet-50. It has been applied successful to address multiple tasks. These tasks include image classification [98, 99, 100, 101] and object detection [102, 103]. It has been applied for new tasks such as Panoptic Segmentation [102] where each pixel of the entire image is classified to one out of several classes [104]. In addition to that, ResNet-50 has been successfully applied to table detection [12, 13] task as well.

Thus, we utilize the ResNet-50 architecture to act as the feature extractor in this work. ResNet-50 was initially designed to address the image classification task. Thus, in Fig. 3.1 we display the functioning of the ResNet-50 architecture towards solving this task. It is worth noting that due to its versatility and flexibility a large section of the architecture can be used for multiple image processing tasks as well. As shown in Fig. 3.1 the ResNet-50 architecture consists of four types of convolution blocks each with different number and type of filters per convolution layer. We refer to these fours block as Conv_2, Conv_3, Conv_4 and Conv_5. We denoted the convolution layers with conv text in the image. These block are repeated $3times, 4times, 6times$ and $3times$ respectively to develop 48 layers of the ResNet-50 architecture. The remaining two layers are the initial convolution layer at the start of the architecture and the Fully Connected Layer which provides the final logits. These logits are processed by the Softmax function which assigns the image class label to the input based on the highest probability. The pooling layers (max_pool and Average Pooling) do not store any learning weights as they are used to pool the feature maps at two sections of the architecture. Hence, they are not counted as a layer when the number of layers of an architecture are being counted.

Furthermore, within each convolution block we also display the skip connections with dotted arrow lines. As shown in each convolution block the skip connection propagates the input by skipping over 3 convolution layers for this architecture. It combines the output of these layers with the input to the layers. This combined output from the convolution block is considered as the input to the next convolution block. The convolution layers have a fixed receptive field that operate as a sliding window over each section of the image. They provide rich features to address image processing tasks such as image classification and object detection. However, they cannot capture all the features that are present with

different scales and orientations due to the fixed receptive field. The sliding window of a standard convolution operation (C) which we refer to as the sampling matrix has a set of weights (w). It samples and stores different sections of the image / input feature map (x) to generate an output feature map (y).

$$y(p_0) = \sum_{p_i \epsilon C} w(p_i).x(p_0 + pi) \tag{3.2}$$

As shown in (3.2), p_0 denotes the central location of the sampling matrix while p_i denotes the neighbouring locations on the matrix. The weights (w) of this matrix at each location are updated when the model is trained to improve it's output feature map representation. However, this sampling matrix does not have any provisions to capture the inputs at multiple scales and orientations. Hence, to resolve this rigidity, Dai et al. [55] proposed utilizing deformable convolution which has an adaptive receptive field to capture variances in feature maps. As shown in (6.1) in addition to a standard convolution operation a deformable convolution operation [55] captures offsets (Δp_i). These offsets store the weights at each location of the sampling matrix when the inputs have multiple scales and orientations. Deformable CNN have shown great success when applied to solve tasks such as table detection [50] and table structure recognition [56].

$$y(p_0) = \sum_{p_i \epsilon C} w(p_i).x(p_0 + pi + \Delta p_i) \tag{3.3}$$

.

Each convolution block as shown in Fig. 3.1 contains three convolutions layers. The deformable convolution layers replace standard convolutions only in the second layer. The first and third layer of each block continues to be a standard convolution. Furthermore, an additional convolution captures the offsets from the features map of the first layer and propagates these offsets with the second layer output to the third layer. As tables in images appear with multiple scales and orientations. Furthermore, the rows and columns withing such tables also appear with different scales even within the same images. Thus, we utilized a ResNet-50 architecture with deformable convolution layers as a feature extractor to address both table detection with the TableDet models as well as table structure detection tasks with the TableStrDet model in datasheet (document) images.

3.1.2 Feature Pyramid Network

The convolution neural network architectures such as ResNet-50 downsample the input image to generate feature maps for tasks such as image classification and object detection. The downsampled feature maps have low resolution but high semantic value that is essential to address these two tasks. The task of object detection involves identifying the object in the image and providing details about its location in terms of four co-ordinates or bounding boxes. We refer to this process as localization. To perform this process object detection models can only work on the low resolution feature maps provided by the feature extractor. Lin et al. [86] presented an additional architecture to provide more low resolution but high semantic value feature maps along with the final feature map to improve the localization performance of object detection models. This architecture is refered to as Feature Pyramid Network. It has successfully been applied to address multiple types of issues. These issues are not limited to detecting small objects [105, 106, 107] and detecting objects in medical images [108, 109] but it has also been applied to further improve object detection with IOU based regression losses [110]

In Fig. 3.2 we display the functioning of two components of an R-CNN based object detection model; Backbone (feature extractor) and FPN. As shown in the Fig. 3.2 the backbone follows the bottom up pathway where the input image is provided from the bottom. The ResNet-50 architecture utilizes four types of convolution blocks namely; Conv_2, Conv_3, Conv_4 and Conv_5. These convolution blocks are repeated 3 times, 4 times, 6 times and 3 times respectively as shown in Fig. 3.2. After each set of repeated convolution blocks a feature map is extracted by the FPN. These feature maps are the results of downsampling the image. We denote these downsampled feature maps as Downsampled_FM_2, Downsampled_FM_3, Downsampled_FM_4 and Downsampled_FM_5 for each set of repeated convolution blocks respectively. These downsampled maps are $1/4, 1/8, 1/16$ and $1/32$ of the height (H) and width (W) of the input image and have $256, 512, 1024$ and 2048 channels respectively.

Downsampled_FM_2 feature map has high resolution but low semantic value on the contrary Downsampled_FM_5 has high semantic value and low resolution. Thus to prove more richer feature maps that have high resolution as well as high semantic value, the FPN combines these feature maps by using $1x1$ convolutions with 256 filters on each of these maps. Thus, it generates new feature maps with the same scale ratio as the downsampled feature maps but with 256 channels. It combines these maps from the top to the bottom as shown in the top down pathway. It generates four new feature maps that are upsampled fromt the downsampled feature maps. We denote these maps as Upsampled_FM_5 to Upsampled_FM_2. To generate output feature maps the FPN then passes the upsampled

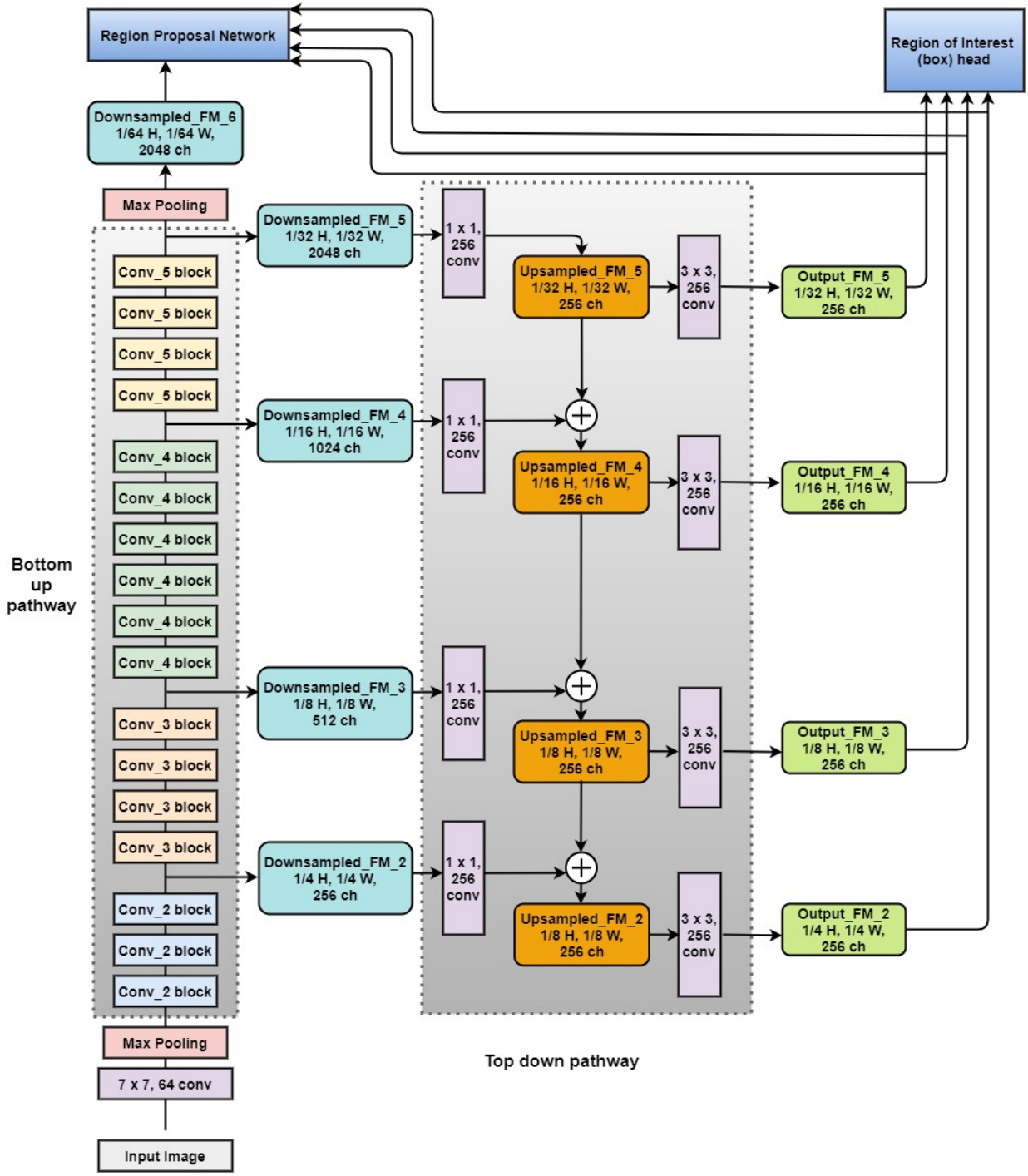


Figure 3.2: **Feature Pyramid Network**

feature maps through $3x3$ convolutions while maintaining the same channel size. This step is taken in order to prevent the upsampled feature maps from facing any aliasing issues.

These four output feature maps ranging from Output_FM_5 to Output_FM_2 are then sent to the other the Region Proposal Network (RPN) and ROI head or box head of the R-CNN architecture. Along with these high semantic and resolution feature maps from FPN the backbone feature map (Downsampled_FM_6) is also sent to these two components as shown in Fig. 3.2.

3.1.3 Region Proposal Networks

Single stage object detection architectures such as Yolov4 [46], SSD [111] and RetinaNet [112] generate proposal bounding boxes without the help of an external convolution network. One stage detectors have simple architectures however, they are outperformed by two or multi staged object detection architectures in terms of detection accuracy [113]. Two or Multi staged object detection models such as Faster R-CNN [45], Mask R-CNN [52] and Cascade R-CNN [11] utilize a Region Proposal Network (RPN) to generate proposal bounding boxes for each object in an image. The RPN aides such networks by filtering out most of the negative proposals. It provides bounding boxes with high confidence scores after multiple steps of refinement and furthermore also prevents any class imbalance between negative and positive bounding boxes [113].

The RPN utilizes ground truth bounding boxes to generate and refine its proposal bounding boxes. It utilizes a convolution layer to generate objectness logits map for each of the five feature maps that the RPN receives as shown in Fig. 3.2. The objectness logits maps lays out the probable location of each object. It utilizes the Output_FM_2 and Output_FM_3 maps to locate smaller objects while other output maps are used to locate larger objects due to their higher semantic value. Furthermore, a large set of multi scaled anchor bounding boxes are generated for each image. Only the anchor boxes that appear with certain Intersection Over Union (IOU) [114] value over the ground truth boxes for the input image are used for refinement. Anchor boxes with greater than 70% IOU value are considered as foreground boxes and a convolution layer is used to generate anchor deltas for refining the boxes [45]. Anchor boxes below 30% IOU value are considered as background while the rest of the anchor boxes are ignored. The RPN performs multiple resampling steps for the foreground (positive) and background (negative) bounding boxes so that it can maintain a balance between the two types of boxes. After applying Non-Maximum Suppression it sends 1000 foreground boxes for refinement to the ROI (box) head. Thus it ensures that only the boxes which appear to match the ground truth bounding boxes are sent for further processing.

3.1.4 ROI (box) head

The ROI (box) head was introduced by Girshick [115] with the Fast R-CNN object detection model. faster R-CNN [45] improved on the Fast R-CNN architecture by utilizing an RPN to provide fewer but more accurate proposal bounding boxes to the box head. Further improvements were made on the type of input the box head received with the introduction of the FPN [86] by He et al. [52] for the Mask R-CNN architecture. Along with the final feature map from the backbone, similar to the RPN, the box head also receives four additional feature maps from the FPN that have high semantic value and resolution. The box head in Faster R-CNN and Mask R-CNN process the proposal bounding boxes and the feature maps in the same format. Both architectures use an ROI Pooling layer to crop rectangular regions that are specified by the proposal bounding boxes.

ROI Align layer was replaced ROI Pooling in the Mask R-CNN architecture [52] in order to provide more accurate regions of interest from the feature maps. The foreground and background proposal boxes are then mapped onto the regions of interest. The proposal boxes that align with ground truth boxes and achieve greater than 50% IOU value are considered as final bounding boxes for the obejct while the rest are background objects. A softmax estimator will assign a label to each box based on the available classes in the ground truth labels. Furthermore, a bounding box regressor will calculate the delta values between ground truth and proposal boxes. The resulting boxes from this regressor are known as predicted bounding boxes. The delta value for regression are calculated by a loss function such as $l1$ loss while the object label classification loss is calculated by cross entropy loss. In addition to these steps, Mask R-CNN [52] also performs instance segmentation where each pixel of the detected object is labeled with the object class.

Cascade R-CNN further enhances the Faster R-CNN and Mask R-CNN architecture by utilizing the box head multiple times. As shown in Fig. 3.3 the architecture uses three box heads successively. The first box head (Box Head 1) refines proposal bounding boxes from the RPN with greater than 50% IOU value. It then sends these boxes as proposals to the second box head (Box Head 2) which refines the proposals at 60% IOU value. The last box head (Box Head 3) receives these improved boxes from Box Head 2 and further refines the boxes greater than 70% IOU value. The final predicted bounding boxes from this architecture are enhanced at a much higher IOU value than the boxes provided by Mask R-CNN and Faster R-CNN architectures. Thus, this architecture has been successfully applied to address multiple tasks such as [14, 47, 48, 116, 117, 118]. Furthermore, the name Cascade was also derived from the cascading machine learning approaches such as Cascading Decision Tree (CDT) [119]. In CDT the lower nodes of of the decision with a large depth provides more accurate results than the nodes higher up the tree, similar to

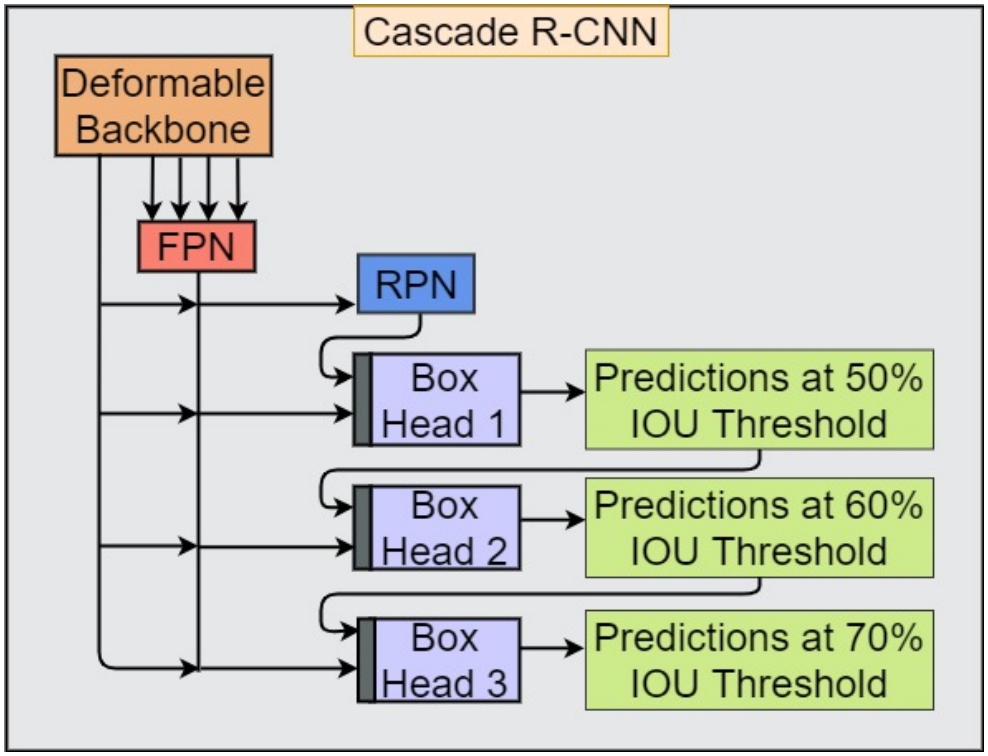


Figure 3.3: **Cascade R-CNN**

the cascading box heads of the Cascade R-CNN architecture [11] where the lowest and last box head provides results at 70% IOU value.

All three box heads the receive one feature map from the backbone and four feature maps from the FPN. Thus, we utilize this architecture to develop both of our solutions; TableDet and TableStrDet to detect tables and table structures respectively. Both solutions address these tasks as object detection tasks. The Cascade R-CNN architecture provides an option to perform instance segmentation with additional convolution layers similar to the Mask R-CNN architecture. We determined that addressing table detection and table structure detection in images as an object detection task was more efficient and accurate than addressing them as an instance segmentation task. Hence, in this work we did not utilize the instance segmentation functionality of the architecture.

3.1.5 IOU-based regression losses

Cascade R-CNN uses three box heads to refine the bounding boxes for each image [11]. Each box head produces bounding boxes with their respective object class labels and confidence scores to denote location and class of each object. These predictions are evaluated in terms of Intersection Over Union (IOU) [114] by measuring the area covered by the predicted / proposed (B) and ground truth (B^{gt}) boxes as shown in (6.2). The boxes are represented with four co-ordinate points. The proposed / predicted bounding box are denoted as $B = (x_1, y_1, x_2, y_2)$ while the ground truth bounding boxes are denoted as $B^{gt} = (x_1^{gt}, y_1^{gt}, x_2^{gt}, y_2^{gt})$. The IOU value is a percentage value that represents how well the two boxes overlap each other thus functioning as a performance indicator for object detection solutions. On the other hand, to measure the regression loss between the two types of boxes traditionally $l1$ loss would be utilized.

$$IoU(B, B^{gt}) = \frac{B \cap B^{gt}}{B \cup B^{gt}} \tag{3.4}$$

$$L_{GIoU} = 1 - IoU + \frac{C - B \cup B^{gt})}{C} \tag{3.5}$$

Since the loss calculation ($l1$) and evaluation metric (IOU) were different an IOU-based loss function [120] was introduced that would match the IOU evaluation metric. However, it did not function for cases where the predicted and ground truth boxes did not overlap and hence Generalized IOU (GIOU) loss [121] was proposed to solve this issue. As shown

in (3.5) GIOU loss generates a large polygon denoted as C that covers both B and B^{gt} and tries to reduce the area of this box so that the predicted and ground truth boxes align with each other.

However this loss does require more time to converge as it tries to refine the overall area of all three boxes at the same time. Furthermore, it also degrades to IOU loss for cases where the predicted box covers the entire ground truth box as well as the reverse case. Hence, Distance IOU (DIOU) loss was introduces by Zheng et al. [18] as a quicker alternative. Instead of calculating the area of the polygon that covers the predicted and ground truth boxes, DIOU loss as shown in (3.6) calculates two distance values. One value is the distance between the center points ($\rho(.)$) of the two boxes denoted by b and b^{gt}. The second value the diagonal length of the polygon that can cover both boxes denoted by c. Utilizing these two parameters instead of calculating the area improves convergence rate of object detection models and also helps inclusion of horizontally and vertically oriented boxes.

$$L_{DIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} \tag{3.6}$$

In datasheet images, tables, figures and text can appear with different scales and orientations. This is also true of table components such as rows and columns. Hence using geometric factors such as distance between center points of bounding boxes could help in effectively detecting objects of full page datasheet images as well as table components. Another geometric factor that could be considered is aspect ratio of the image and objects within it. However, DIOU does not consider the aspect ratio of the bounding boxes in its calculation. As all objects in full page datasheet images as well as components such as rows and columns can appear at different scales, including the aspect ratio in the loss function calculation could provide more accurate detection results.

$$v = \frac{4}{\pi^2}(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h})^2 \tag{3.7}$$

$$\alpha = \frac{v}{(1 - IoU + v')} \tag{3.8}$$

$$L_{CIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \tag{3.9}$$

To consider the scale of an object during loss calculation Zheng et al. [18] developed a second loss function, named as Complete IOU (CIOU). In addition to distance based parameters of DIOU loss it also considers the aspect ratio of the bounding boxes in its calculation. The CIOU loss function adds an additional parameter to measure the consistency of aspect ratio as shown in (6.4) denoted as v and a positive trade-off parameter denoted as α as shown (6.5). The new parameters introduced by CIOU loss can be added to DIOU loss as shown in (6.6). In this work we utilize CIOU loss at each of the box heads in the TableDet models and the TableStrDet model that address table detection and table structure detection respectively.

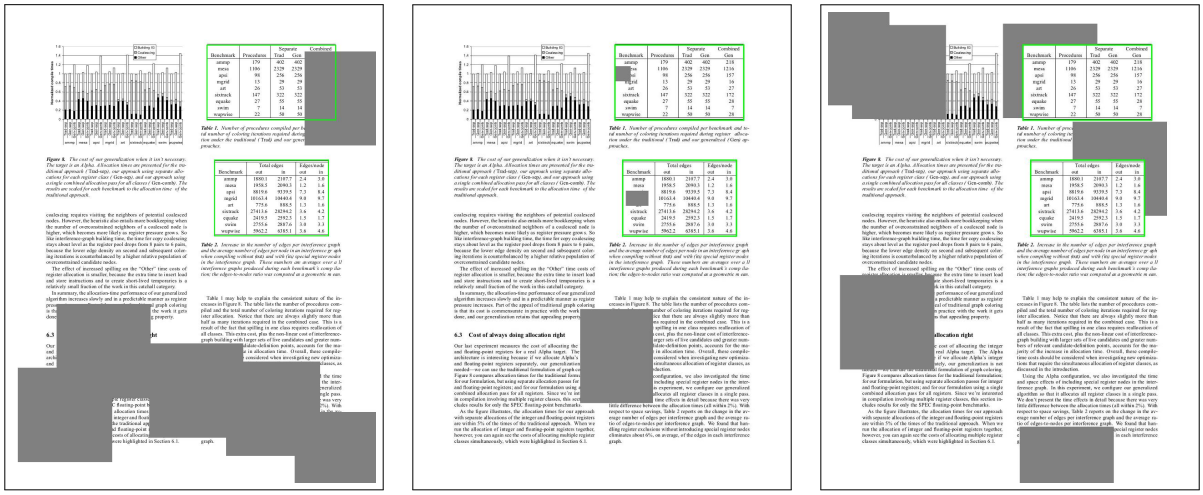
3.2 Data Augmentation

Object detection architectures such as Cascade R-CNN require large amounts of annotated (locations of object) data in the form of bounding boxes to learn the location and types of objects in the image. Such complex artificial neural network based models are prone to overfitting when the size of the training data is not large and diverse [122]. Multiple image augmentation strategies have been presented to improve the functionality of image classification architectures such as DropConnect [123], Random cropping, image mirroring [124, 29] and scale jittering [125]. Furthermore, certain augmentation strategies apply patches of noise to the image to improve the accuracy [126, 127, 128, 129] and robustness [130, 131] of the image classification architectures.

With multiple such augmentation strategies available for image classification architectures to improve their accuracy recent works such as AutoAugment [132] and Fast AutoAugment [133] have investigated the impact of letting these architectures learn the best set of augmentations that can improve their accuracy on a particular dataset. Both works set up a search space to identify which set of augmentation strategies along with the magnitude and prop ability of applying the strategy can help the architecture improve its output feature representation. Learned Augmentation [134] is one of the first techniques to apply this search space augmentation strategy for object detection architectures. It considers the impact of scaling, shifting, rotating the image. It also learns the impact of these strategies along with adding patches of noise within the bounding boxes and the objects surrounding the bounding boxes.

The patches of noises which are also refereed to as occlusion are proven to be an effective technique in preventing the deep neural network architectures from overfitting and making them generalizable to unseen test data as shown in [129, 126, 128]. All three augmentation strategies [129, 126, 128] place N number of squares / masks over sections of the image.

This forces the architecture to pay more attention to other sections of the image so that it can generalize better to the global context of the image.



(a) Random Occlusion (b) Random Erase (c) Table Aware Cutout

Figure 3.4: **Example of Random Occlusion with $N = 6$ squares, Random Erase with $N = 3$ rectangles (1 for each table and 1 from the image) and Table Aware Cutout**

Random Occlusion [129] and Random Erase [128] are two occlusion strategies that aim at improving image classification and object detection models respectively. As shown in Fig. 3.4(a), Random Occlusion places N number of masks / patches randomly across the image without any restrictions on where the masks are placed or whether the masks overlap with each other. The Image and Object aware Random Erase strategy as shown in Fig. 3.4(b) applies a single mask to the entire image and individual masks to each object of the image. Both techniques are proposed for images with natural objects. However, in the case of tables in datasheet (document) images such augmentation strategies have to be tailored to help the model differentiate between text, figures and tables objects. Hence, we introduce a new augmentation strategy called Table Aware Cutout (TAC).

An image in a mini-batch of training data is selected for TAC with a probability p. TAC applies N masks randomly across the image as show in Fig. 3.4(c). However, unlike Random Erase [128], TAC is applied around the table without obstructing or occluding any part of the table. Thus, the masks near the table do not overflow onto the table region but rather reveals the edges of the table. Furthermore, the masks that are further away from the table are applied without any restrictions. Each pixel in each mask is set to a constant value V.

This strategy is only applied when the TableDet models is trained to detect tables, text and figures in full page images. At every instance when an image with a table object is picked to be augmented with TAC, the masks are randomly placed on the image. For instance, if the masks are applied around the table region then the model will be able to focus on all the unobstructed regions which will include the table and other objects. If some masks are applied near the table the model will be able to distinguish between the table text and surrounding objects. Thus, every time a TAC augmented image is shown to the model it will help the model focus its attention on a different section of the same image.

Thus, preventing the model from overfitting on any local context of the image such as the table region. Rather it helps the model understand the context around the table region as well. In addition, TableDet models can detect tables, text and figures are thus are trained to learn the global context of an image. Utilizing TAC augmented images can enhance the model's understanding of the global context and furthermore enhance its table detection performance as we present in Chapter 4.

3.3 Evaluation Metric

In this work we propose TableDet in Chapter 4 which solves table detection and table image classification as an object detection and image classification task. We also propose TableStrDet in Chapter 6 which address table structure detection as an object detection task. When the task is solved as an image classification we evaluate and compare our results with Recall, Precision and F1 score as shown in (3.10), (3.11) and (3.12) respectively.

$$Recall = \frac{TruePositive(TP)}{TruePositive(TP) + FalseNegative(FN)} \tag{3.10}$$

$$Precision = \frac{TruePositive(TP)}{TruePositive(TP) + FalsePositive(FP)} \tag{3.11}$$

$$F1 - score = 2 * \frac{Recall * Precision}{Recall + Precision} \tag{3.12}$$

To evaluate our object detection results for table detection and table structure detection we utilize the COCO [114] evaluation metric. It utilizes the IOU evaluation metric. An IOU value as shown in (6.2) measures the overlapping area shared between the ground

truth and predicted bounding boxes. Thus it can be measured at 10 potential overlap area percentages ranging from 50% to 95%. For instance at 50% IOU if the boxes share an overlap area of more than 50% then the predicted bounding box is considered a True Positive else it is considered as False Positive. If the predicted box does not overlap over any ground truth box the it is considered as False Negative. The COCO evaluation metric utilizes the True Positive, False Positive and False Negative values to calculate the recall and precision at each IOU value.

$$mAP(AP) = \frac{AP^{50IOU} + AP^{55IOU} .... + AP^{95IOU}}{10} \tag{3.13}$$

$$mAR(AR) = \frac{AR^{50IOU} + AR^{55IOU} .... + AR^{95IOU}}{10} \tag{3.14}$$

These values will be averages of recall and precision at specific IOU values thus we refer to these values as Average Precision and Average Recall. However, to get a single value which represents how well the object detection solution is performing at all IOU values the average recall and average precision values are averaged to generate mean Average Recall (mAR) and mean Average Precision (mAP) respectively. However, COCO evaluation metrics use the term mean Average Precision and Average Precision. A similar case is noticed for mean Average Recall and Average Recall [114]. Thus in this work we utilize Average Recall and Average Precision to measure the capability of our table detection and table structure detection solutions.

Chapter 4

Table Detection

The task of table detection has been well investigated from multiple perspectives. These perspectives change based on the type of input used by different solutions in order to address this task. Previous works have either preferred to directly identify the tables in the PDF documents or have converted the document pages into images and have then identified tables in each image. PDF based solutions benefit from the availability of the meta data of PDF document to detect the table and its contents [7, 135]. However, this also restricts such solutions from being applied to documents that have an organized meta data [135]. On the other hand utilizing an document page as an image provides solutions with the freedom to generalize to all types of tables in all types of documents. These images can also include scanned images which would not be considered by PDF based table detection solutions.

Furthermore, the application of deep learning-based solutions (models) have achieved significant performance improvements in tasks such as object detection [48, 116] and image segmentation [136, 137]. This has prompted an active interest in the domain of document analysis to utilize such solutions to identify and recognize tables and their structures. Thus, recent works have focused more on identifying tables in document or datasheet full page images [15, 32, 50, 12, 13, 17, 14]. These methods do provide encouraging results despite the disadvantage of not having access to the meta data of the documents. They solve the task of table detection as an object detection task on images. They use Intersection over Union (IOU) [114] to compare and evaluate their table detection performance which is commonly used to compare object detection performances of different solutions.

However, these solutions only detect tables and do not detect other objects such as figures and text in the image nor do they analyze the impact of detecting these objects.

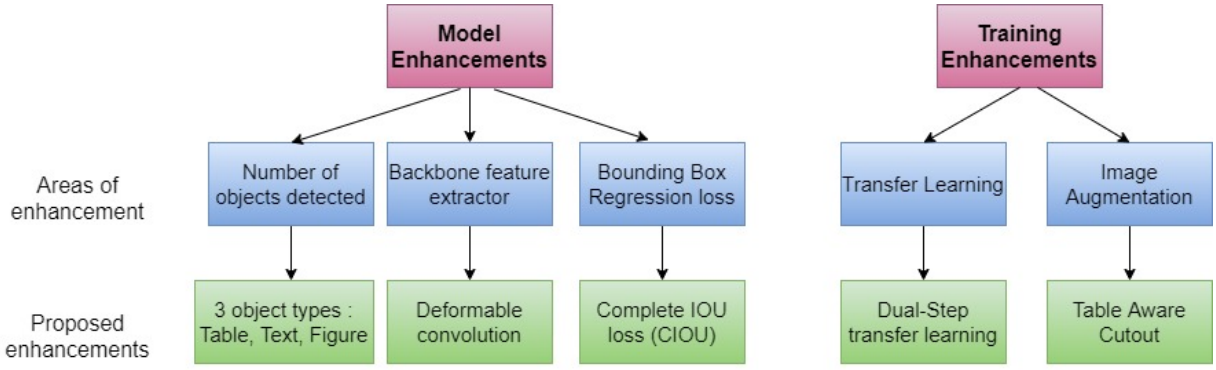


Figure 4.1: **The enhancements proposed by the TableDet approach to solve table detection and table image classification. The green boxes indicate our proposed enhancements.**

Thus, we propose TableDet, a deep learning-based model that can accurately detect and classify an object in a document or datasheet image as a table. We propose detecting text and figure objects along with table objects in datasheet images to improve the table detection performance of the TableDet model. Thus, the TableDet model can classify an object into one of the three classes: table, text and figure. We display the improvements brought on by the utilization of a deformable convolutions [55] backbone to extract multi-scale and multi-oriented features from the image. In addition we display the performance improvement provided by utilizing Complete IOU loss [18] for bounding box regression to handle the multi-scale variations in table sizes. In addition to the deep learning architecture enhancements we provide details on the training enhancements that help TableDet improve its table detection performance. These training enhancements include a dual-step transfer learning process and a new image augmentation strategy called Table Aware Cutout (TAC) that helps the model detect tables at higher IOU values. As shown in Fig. 4.1 we show the model and training enhancements proposed by this approach.

We describe the steps taken to develop the TableDet model, the proposed training approach and augmentation strategy tailored for table detection solutions in Section 4.1. We describe the public and private datasets utilized in our analysis in Section 4.2 and the training details in Section 4.3. We display our step wise analysis that we take to develop the TableDet model and the impact of our enhancements in Section 4.4. We also provide our table detection performances on the public and private datasets in Section 4.4. We provide our concluding statements in Section 4.5

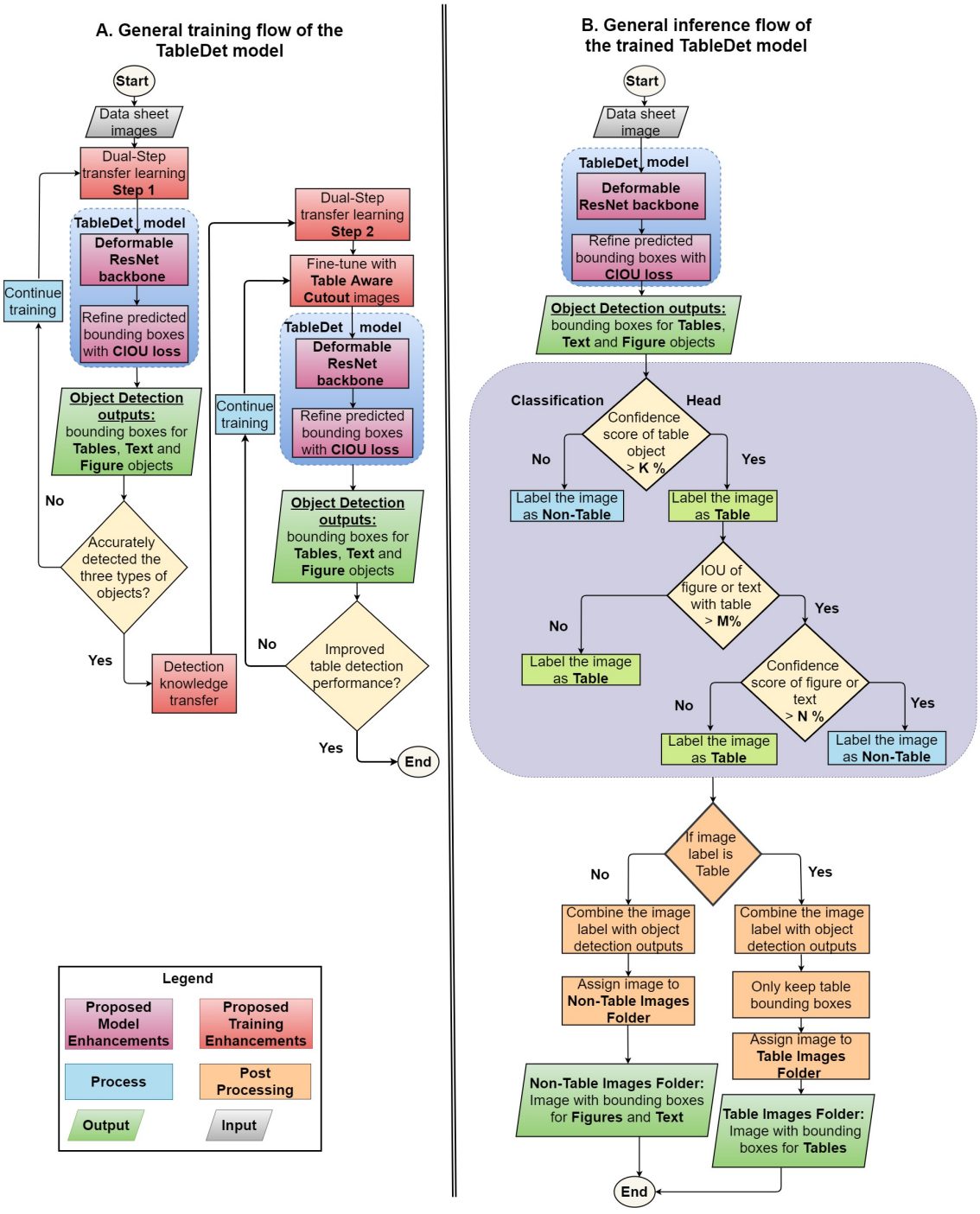


Figure 4.2: **TableDet: Training Process (A) and Inference Process (B)**

4.1 System Model

In this section, we describe the proposed approach to address the table detection task in datasheet (document) images. We put forward, the TableDet model, a deep learning-based approach that detects tables, text and figures in document and datasheet images. We describe the model and training enhancements are utilized to develop the TableDet model and to improve its table detection performance.

4.1.1 Model architecture

Deep learning methods have excelled in the task of object detection on public datasets such as COCO [114] and ImageNet [138]. These methods are not only robust in detecting large number of natural objects but also in identifying variations in objects of the same class [11, 10]. As tables in datasheet images can appear in various scales, orientations and layout we propose solving the task of table detection as an object detection task utilizing a deep learning approach. A universal table detection solution would have to handle high intra and low inter class variances to accurately detect tables in images. The high intra class variance is caused by the different table structures that appear in such images. These differences occur due to multiple factors such as presence or absence of line separators between cells. It also arises due to the presence of merged cells (a common cell shared by multiple neighbouring cells) [21]. The merged cells present the information in the table in a hierarchical layout. On the other hand, the low inter class variance is caused by similarities shared between tables, text and figures in such images. Thus, we propose, TableDet, a deep learning-based approach to detect tables in datasheet (document) images.

We employ the Cascade R-CNN [11] architecture to develop the TableDet model. The Cascade R-CNN architecture as shown in Fig. 4.3 consists of the backbone, region proposal network (RPN), Feature Pyramid Network (FPN) and a sequence of box heads. The TableDet model considers the multi-scale and multi-orientated nature of all objects in datasheet (document) images, especially tables, to improve its performance in detecting tables. In this work, we use a ResNet model as a backbone to extract rich features from an image. The ResNet architecture comprises of a sequence of convolution layers stacked on top of each other in a format that can extract features in the most efficient manner. These layers reside in blocks and extract features at multiple scales as shown in Fig. 3.1. We utilize a FPN to extract these multi-scale features as shown in Fig. 3.2. However, convolution layers have a fixed receptive field while deformable convolution layers [55] have an adaptive receptive fields. The adaptive receptive field can adapt to capture variations

in scales and orientations of tables in datasheet (document) images. Thus, along with the FPN we utilize deformable convolutions in the backbone of the TableDet architecture.

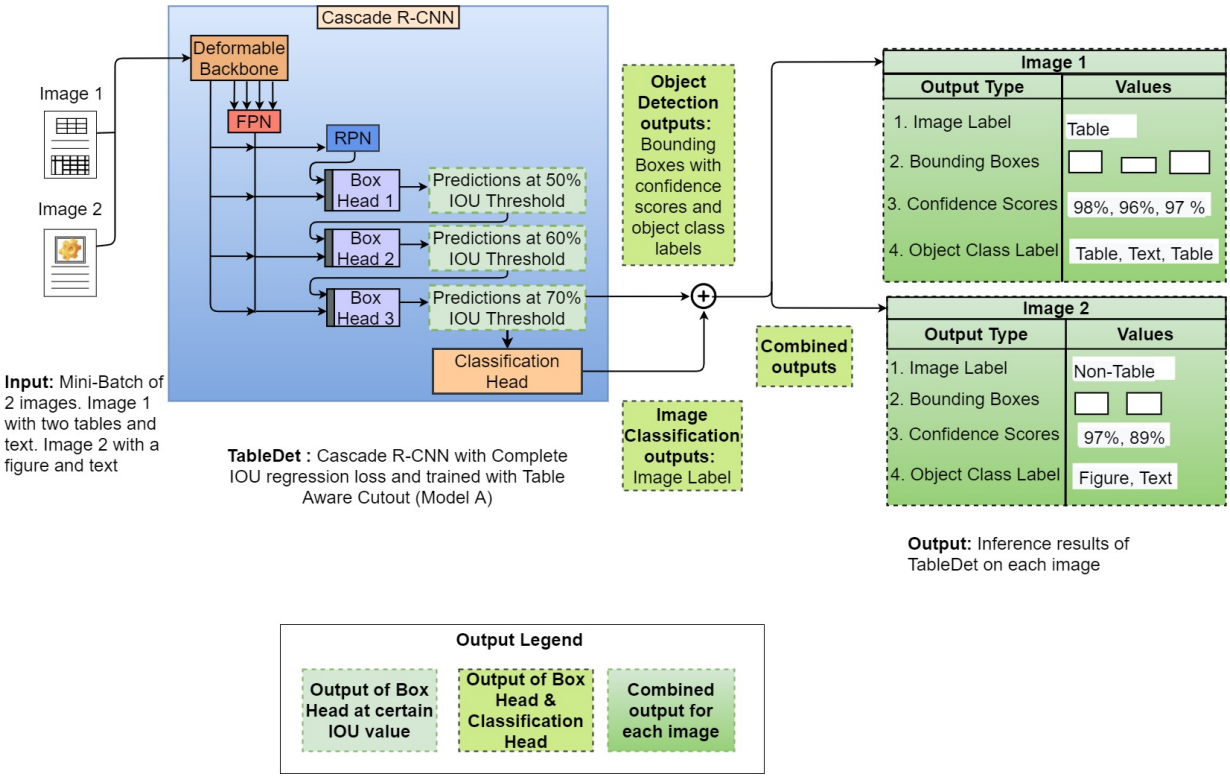


Figure 4.3: **TableDet (Model A): Inference of the trained model**

The deformable convolution backbone and the FPN provide rich multi-scale features to the RPN and sequence of box heads as shown in Fig. 4.3. At the RPN the proposal bounding boxes are generated using the ground truth bounding boxes as references and then sent to the sequence of box heads. Cascade R-CNN utilizes three box heads to refine the bounding boxes for each image [11]. Each box head receives the proposed / predicted bounding boxes from the previous box head except for the first box head. In addition they also receive multi-scale feature maps from the FPN and final layer of the backbone. Each box head refines the predicted bounding boxes at higher Intersection Over Union (IOU) [114] values. An IOU value is the area covered by the predicted (B) and ground truth (B^{gt}) boxes as shown in (6.2). The first box head receives the proposal bounding boxes from the RPN. It refines the boxes that achieve greater than 50% IOU value and sends them for refinement to the second box head. At the second and third box head the predicted

boxes greater than 60% and 70% IOU values respectively are refined. Each box head also provides the object class labels and confidence scores of each predicted bounding box.

$$IoU(B, B^{gt}) = \frac{B \cap B^{gt}}{B \cup B^{gt}} \tag{4.1}$$

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{4.2}$$

As shown in (6.2) the IOU values is the intersection over union of proposed / predicted bounding box $B = (x_1, y_1, x_2, y_2)$ and ground truth bounding box $B^{gt} = (x_1^{gt}, y_1^{gt}, x_2^{gt}, y_2^{gt})$. To measure and improve the quality of bounding boxes a multi-task loss is calculated at each box head. As shown in (6.3) we compute classification loss for object class labels and regression loss for bounding box values together. Conventionally $l1$ loss and cross entropy loss are used to measure the regression loss and classification loss respectively. However, since an IOU value is used to evaluate the performance of each box head we propose utilizing the Complete IOU loss [18] to match the evaluation and regression loss calculation metrics. In addition Complete IOU (CIOU) loss [18] utilizes the same parameters of Distance IOU (DIOU) loss [18]. DIOU loss in turn is a faster alternative of Generalized IOU [121] loss. There are two common parameters shared between CIOU and DIOU loss. First, the distance between the center points ($\rho(.)$) of the two boxes denoted by b and b^{gt}. Second, the diagonal length of the polygon that can cover both boxes denoted by c. These parameters improve convergence rate of object detection models and also helps with the inclusion of horizontally and vertically oriented boxes.

$$v = \frac{4}{\pi^2}(\arctan \frac{w^{gt}}{h^{gt}} - \arctan \frac{w}{h})^2 \tag{4.3}$$

$$\alpha = \frac{v}{(1 - IoU + v')} \tag{4.4}$$

$$L_{CIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \tag{4.5}$$

The CIOU loss function adds an additional parameters to measure the consistency of aspect ratio as shown in (6.4) denoted as v and a positive trade-off parameter denoted as α as shown (6.5). The new parameters introduced by CIOU loss can be added to DIOU loss as shown in (6.6). In datasheet (document) images, tables, figures and text can appear in different orientations and scales. Hence using geometric factors such as distance between center points of bounding boxes could help in effectively detecting objects of all three classes. In addition to that, another geometric factor that could be considered is aspect ratio of the image and objects within it. As all three object classes in datasheet (document) images can appear at different scales even within the same image including aspect ratio in the loss function calculation could provide more accurate detection results. Thus, we propose utilizing CIOU loss at all three box heads to improve the bounding box regression performance for all three types of objects.

Furthermore, we propose that along with detecting tables in the image that the model should also detect the text and figure objects. We propose classifying each object in the datasheet image into one of the three classes: table, text and figure. By classifying the detected object into one of the three classes the model can learn about the structure and layout of these objects in the datasheet (document) image. Furthermore, differentiating between these objects can help the model identify the differences between these three types of objects. Thus unlike previous table detection solutions, TableDet not only learns to identify tables in a local context but also learns the important global context of datasheet (document) images which helps it accurately distinguish objects of each of these three classes.

The object detection outputs for each input image of the TableDet model are the detected bounding boxes with their respective object class labels and confidence scores as shown in the inference results in Fig. 4.3. We also determined that detecting tables, text and figures in the images also helps the model classify full page datasheet (document) images into two classes, those that have tables and those that do not. We introduce a classification head as shown in Fig. 4.3 into the model architecture to address this task which we refer to ask table image classification. We describe the process that the TableDet model utilizes to generate image classification outputs in more detail in Chapter 5.

Thus, we propose a table detection enhanced model, TableDet, which utilizes a deformable convolution backbone, Complete IOU (CIOU) loss at each box head and classifies each object in the datasheet (document) image as a table, text or figure. To show the impact of using deformable convolutions over conventional convolutions in the backbone architecture we developed two model variants of TableDet in this work. Both variants use CIOU loss in each of the three box heads but have different backbones. We refer to the TableDet model with deformable convolution ResNet backbone as Model A and the

model with conventional convolution ResNet backbone as Model B. These models have been constructed with the help of the Detectron2 library [1] .

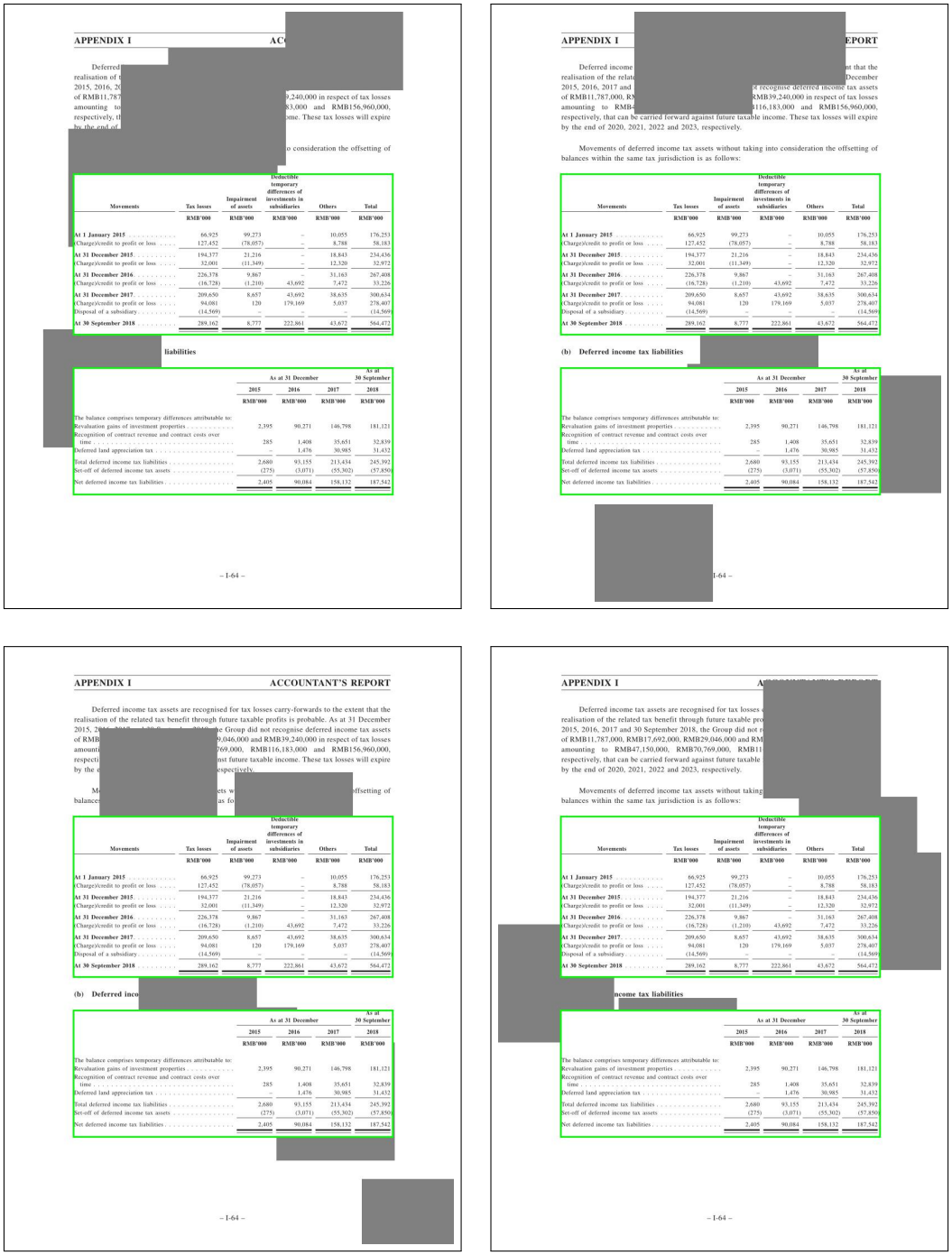
4.1.2 Image Augmentation : Table Aware Cutout

As TableDet solves the task of table detection as an object detection task and utilizes and deep learning-based architecture it is also prone to overfitting like other object detection architectures [122]. There are many image augmentation / transformation strategies such as scale jittering [125] and occlusion [129, 126, 128] that can prevent object detection architectures from overfitting on the training data. Scale jittering provides the same image at different scales to the model while it is being trained. The three occlusion methods [129, 126, 128] on the other hand place N number of squares / masks over sections of the image. However, these methods have proven to be effective on natural images with naturally occurring objects which are different from objects that appear in datasheet (document) images. Hence, we introduce a new augmentation strategy called Table Aware Cutout (TAC) that is tailored towards improving the table detection performance of the TableDet model.

TAC is an optimized table aware occlusion strategy that places N masks over a datasheet (document) image. TAC is applied around the table without obstructing or occluding any part of the table. The masks near the table do not overflow onto the table region of the image but rather they reveal the edges of the table. The masks that are further away from the table are applied without any restrictions as shown in Fig. 4.4. Each pixel in each mask is set to a constant value V. This strategy is applied on one image per mini-batch during training with a probability p. At every instance that an image is picked for TAC augmentation, the model is exposed to a different unobstructed section of the image. As shown in Fig. 4.4 different sections of the same image are exposed to the model when it is trained to prevent it for overfitting on any local context of the image.

For instance, if some masks are near the table the model will be able to differentiate between the table and other objects such as surrounding text. If the masks are away from the table then the model will be able to focus on all the unobstructed regions which will include the table and other objects. Thus, preventing the model from overfitting on any local context of the image such as the table region. Rather it helps the model understand the context around the table region as well. Furthermore, as TableDet models detect tables, text and figures, TAC helps the model improve it's understanding of the global context and furthermore enhances its table detection performance.

[1] https://github.com/facebookresearch/detectron2



APPENDIX I ACCOUNTANT'S REPORT

Deferred income tax assets are recognised for tax losses carry-forwards to the extent that the realisation of the related tax benefit through future taxable profits is probable. As at 31 December 2015, 2016, 2017 and 30 September 2018, the Group did not recognise deferred income tax assets of RMB11,787,000, RMB17,692,000, RMB29,046,000 and RMB39,240,000 in respect of tax losses amounting to RMB47,150,000, RMB70,769,000, RMB116,183,000 and RMB156,960,000, respectively, that can be carried forward against future taxable income. These tax losses will expire by the end of 2020, 2021, 2022 and 2023, respectively.

Movements of deferred income tax assets without taking into consideration the offsetting of balances within the same tax jurisdiction is as follows:

Movements	Tax losses	Impairment of assets	Deductible temporary differences of investments in subsidiaries	Others	Total
	RMB'000	RMB'000	RMB'000	RMB'000	RMB'000
At 1 January 2015	66,925	99,273	–	10,055	176,253
(Charge)/credit to profit or loss	127,452	(78,057)	–	8,788	58,183
At 31 December 2015	194,377	21,216	–	18,843	234,436
(Charge)/credit to profit or loss	32,001	(11,349)	–	12,320	32,972
At 31 December 2016	226,378	9,867	–	31,163	267,408
(Charge)/credit to profit or loss	(16,728)	(1,210)	43,692	7,472	33,226
At 31 December 2017	209,650	8,657	43,692	38,635	300,634
(Charge)/credit to profit or loss	94,081	120	179,169	5,037	278,407
Disposal of a subsidiary	(14,569)	–	–	–	(14,569)
At 30 September 2018	289,162	8,777	222,861	43,672	564,472

(b) Deferred income tax liabilities

	As at 31 December			As at 30 September
	2015	2016	2017	2018
	RMB'000	RMB'000	RMB'000	RMB'000
The balance comprises temporary differences attributable to:				
Revaluation gains of investment properties	2,395	90,271	146,798	181,121
Recognition of contract revenue and contract costs over time	285	1,408	35,651	32,839
Deferred land appreciation tax	–	1,476	30,985	31,432
Total deferred income tax liabilities	2,680	93,155	213,434	245,392
Set-off of deferred income tax assets	(275)	(3,071)	(55,302)	(57,850)
Net deferred income tax liabilities	2,405	90,084	158,132	187,542

Figure 4.4: **Table Aware Cutout example on the same image**

4.1.3 Dual-step transfer Learning

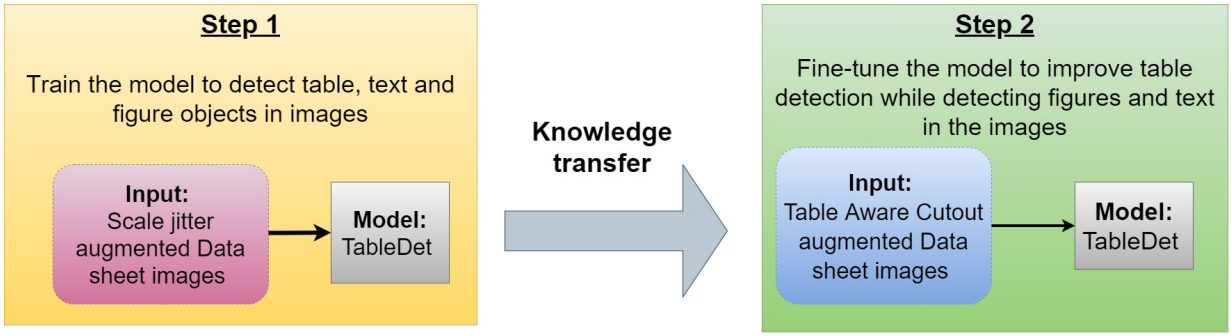


Figure 4.5: **Dual-step Transfer Learning**

Object detection approaches require large number of images such as [114] to train and improve their object detection performances. However, models that are trained on such large datasets as [114] can be trained on datasets that do not have naturally occurring images by the concept of transfer learning. The naturally occurring objects such as dogs and cats do not share a visual similarity with page objects such as tables. However the knowledge about features that the models have learnt from being trained on the natural images such as borders and outlines of objects can be utilized to train the model to detect page objects in document images.

Thus, as per transfer learning the TableDet model with a Cascade R-CNN architecture can be initialized with COCO pre-trained weights and trained on datasheet (document) images to improve its detection performance. In addition, we propose training the model in a dual-step transfer learning process to enhance its table detection performance as shown in Fig. 4.5. The process consists of two incremental steps to train the model to detect tables, text and figures. To train the model we have used the available table detection datasets that have the locations (annotations) of each table in the images. We have also added annotations for the text and figure regions to these images.

At step 1 we train the model to detect tables, text and figures in the datasheet (document) images and to learn about the variations in the format and layout of objects in a datasheet (document) image by detecting objects of these three classes. This step is conforms to the classical definition of transfer learning where the model was trained on one dataset and then the object detection knowledge was used to train it on another dataset that does not have any images similar to the first dataset. At this step, it is trained with scale jitter augmented images to help it learn the features of these types of objects and layout of the content in the image.

In step 2, we fine-tune this model with Table Aware Cutout (TAC) augmented images to further enhance its table detection performance. As shown in Fig. 4.5 during step 1 the model would be trained to differentiate between table, text and figure objects. During the fine-tuning process of step 2 the model can then build on the knowledge it gained from the previous step and improve its table detection performance with the help of TAC augmented images. While the dataset and task in step 1 and step 2 are the same we proposed this additional step of transfer learning to help the model improve its localization capability and to build on it's knowledge rather than training it from scratch. Thus, the dual-step process builds the detection knowledge of the model in an incremental format. Furthermore, this training process reduces the overall time taken to train the model to detect objects of these three classes with an enhanced table detection capability.

4.2 Datasets

In this work, we have utilized four public and one closed document table detection datasets to train and test the proposed Model A and Model B variants of the TableDet model. We utilized the publicly available datasets from ICDAR 2013 [68], ICDAR 2017 [79], ICDAR 2019 [8] and Marmot [2] to develop a public dataset as shown in Table 4.1. We use this public dataset in conjunction with the closed dataset (Lytica) to create another dataset which we refer to as private dataset as shown in Table 4.1. We describe each individual datasets in detail in Section A.1.1. Furthermore, we have added the locations (annotations) of figures and text to the available table object annotations to each image in each dataset.

As shown in Table 4.1 the ICDAR 2013 dataset includes 238 images. It is worth noting that all 238 images are used to test the Model A or Model B variants of the TableDet. Not a single image is used to train or fine-tune the models. The ICDAR 2017 dataset includes 2417 images and we maintain the same training and test split as the official competition [79] in order to fairly compare out results with state-of-the-art solutions. Thus 817 images are utilized for testing while the remaining 1600 images are utilized for training. In this study we utilize the modern document pages from the ICDAR 2019 table detection competition [8] to develop the public and private datasets. We utilize 600 images for training and the official 240 test images for testing. Furthermore, we have utilized the complete Marmot dataset which comprises of 1000 Chinese document images and 1000 English document images to enhance the training set.

[2]http://www.icst.pku.edu.cn/cpdp/sjzy/index.htm

Table 4.1: **Datasets used to train and test TableDet**

Set name	Datasets	Training	Test
Public dataset	ICDAR 2013	-	238
	ICDAR 2017	1600	817
	ICDAR 2019	600	240
	Marmot	2000	-
	Total	4200	1295
Private dataset	Lytica	1000	240
	Public Dataset	4200	-
	Total	5200	240

These four datasets are combined to develop the public dataset as shown in Table 4.1. In addition to this we have also utilized a closed dataset which we refer to as the Lytica dataset. It includes 1240 images of which we have utilized 1000 images for training and 240 for testing. This dataset is used with the datasets from the public dataset to develop the private dataset as shown in Table 4.1. The TableDet model variants (Model A and Model B) are trained on public and private datasets to ensure that we can fairly evaluate the table detection performance with state-of-the-art solutions that have also utilized similar datasets.

4.3 Training Details

In this work, we introduce TableDet, a deep-learning based table detection approach in datasheet (document) images. We introduce two variants of the TableDet model. Both variants utilizes ResNet-50 architecture as a backbone feature extractor. We refer to the TableDet model with deformable convolution ResNet as Model A and the model with conventional convolutions ResNet as Model B. Furthermore, both models also use an FPN to extract feature maps at multiple scales. By adopting transfer learning we use the COCO 2017 pre-trained weights of Cascade R-CNN architecture from Detectron2 to reduce the amount of time required to train the models. While the domain of natural objects in COCO 2017 dataset and tables in table detection datasets are different the object detection knowledge can be transferred as previously shown in [12, 17, 14, 15].

We developed two datasets as shown in Table 4.1 to train two separate copies (sets) of Model A and Model B. Both sets of models are trained on a single NVIDA Tesla V100 GPU with 32GB RAM. We train the models as per our proposed dual-step transfer learning process as shown in Fig. 4.5. The first dataset is the public dataset that consists of all public datasets as shown in Table 4.1. The second dataset is the private dataset that consists of the Lytica dataset and the public dataset. During the training process, the table detection performance of each model is evaluated on a test set. When the models are trained on the public dataset their performance is evaluated on ICDAR 2013 (complete set). When the models are trained on the private dataset their performance is evaluated on the Lytica test set.

As per the dual-step transfer learning process we evaluate the table detection performance on a test set in terms of Average Recall (AR) and Average Precision (AP). As per step 1 of the learning process we train the models with scale jitter augmented images for 20 epochs to detect tables, text and figures. Scale jittering [125] randomly resizes the shortest side of the image to one of the values in this range : $[640, 672, 704, 736, 768, 800]$. We save a version of the model after each epoch during step 1. The model that achieves the highest AR and AP on the test set within these 20 epochs is picked for further training. Next in step 2 we fine-tune the models with Table Aware Cutout (TAC) augmented images for 20 epochs to improve their table detection performance while also detecting text and figures in the images.

TAC is applied with a probability (p) of 0.5. We determined that the number of squares (N) and content value of each square (V) when applying TAC should be $N = 6$ and $V = 128$ to achieve an optimal table detection result. TAC is used in conjunction with scale jitter. We trained the models with a linear learning rate policy and used a learning rate of 0.002 with a SGD optimizer and momentum of 0.9. We trained the models with a small batch size of 2. Thus, instead of utilizing the standard Batch Normalization [139] to normalize the inputs we utilized Group normalization [140]which is better suited for training processes where batch sizes are small. We also used a regression loss weight of 10 which was described by [18] to normalize the results of CIOU regression loss. We do not train the model for the task of table image classification. We train the TableDet model variants only for the table detection task and use their enhanced table detection results to solve the table image classification task at inference time. We describe the table image classification process in more detail in Section 5.1

4.4 Results

In this section, we describe the analysis we have performed to help us to build the TableDet model after utilizing the proposed model and training enhancements. For the model enhancements we propose detecting tables, text and figure objects, utilizing a deformable convolution backbone and utilizing Complete IOU regression loss to improve the table detection performance of the model. For the training enhancements we propose Table Aware Cutout augmentation in conjunction with dual-step transfer learning to train the model efficiently. In Section 4.4.1 we show the impact of using two of the model enhancements and both of the training enhancements. In Section 4.4.2 we describe the performance of the TableDet model variants against state-of-the-art table detection solutions on four test sets.

4.4.1 Preliminary Analysis

In the first stage of our proposed table text extraction pipeline, we have proposed the construction of a model that could accurately detect all tables in datasheet images. This entails accurately locating the table object and correctly classifying the object as a table and not a figure or text. Furthermore, we also propose addressing the table image classification task which we describe in more detail in Section 5.1. In Table 4.2 we show the impact of detecting text and figures along with tables in the datasheet images on the table detection task. The number of classes (No. of object classes) column indicates the number of object types that the model can detect.

Table 4.2: **Table Detection results on ICDAR 2013 (complete set) as a test set with 1 class (table) vs 3 classes (table, figure, text)**

System	No. of object classes	Table Detection		
		Average Recall	Average Precision	Average F1
Cascade	1	99	98.6	98.8
R-CNN	3	99.2	98.8	99

As shown in Table 4.2 we used two Cascade R-CNN models with $l1$ loss to measure the impact of detecting more objects on addressing the table detection task. We evaluated the table detection performance of each model in terms of Average Recall (AR) and Average Precision (AP). Both metrics are calculated over ten IOU values ranging from 50% to 95%.

Furthermore, the models are trained on the training split of the public dataset from Table 4.1. We use the complete ICDAR 2013 dataset to test the generalization ability of the models thus, no image from this dataset is used to train either of the models.

The Cascade R-CNN model that detects only tables (1 class) in the images is considered as our baseline system in this situation. The model that detects tables, text and figures (3 classes) is considered as our proposed system. As shown in Table 4.2 the model that detects tables, text and figures can improve its table detection performance on both AR and AP by 0.2%. Thus, we determined that detecting three classes of objects on the datasheet images can help the model improve its table detection performance even when none of the test images appear in the training set.

Table 4.3: **Table detection results comparison between multiple regression losses on ICDAR 2013 (complete set) as a test set**

System	Loss	Table Detection		
		Average Recall	**Average Precision**	**Average F1**
Cascade R-CNN	$l1$ (Baseline)	99.2	98.8	99
	DIOU	99.3	98.9	99.1
	CIOU	**99.7**	**99.4**	**99.5**

We also analyzed the impact of using IOU-based regression losses in the Cascade R-CNN architecture as shown in Table 4.3. We determined that detecting three classes of objects improve the object detection performance of a model as compared to only detecting table objects in Table 4.2. Thus, we trained three Cascade R-CNN models on the public dataset and evaluated their table detection performance on the complete ICDAR 2013 dataset. We measure the table detection performance of the models in terms of Average Recall (AR) and Average Precision (AP). The first model used the standard $l1$ loss for bounding box regression while the second and third models used Distance IOU (DIOU) and Complete IOU (CIOU) losses respectively.

In this situation the model with $l1$ loss is considered as baseline. As shown in Table 4.3 using IOU based losses outperforms the model using $l1$ loss on both metrics. CIOU achieves the highest performance on both metrics. Thus, we determined a bounding box regression loss that considers distance between bounding boxes as well as aspect ratio of the boxes is a better fit for table detection. As shown in Fig. 4.6 we displayed the improvements in table detection performance provided by these two model enhancements; increase in number of detected classes and IOU based regression loss. Thus, we determined that detecting three classes of objects and utilizing CIOU loss is beneficial for improving the table detection performance of the model.

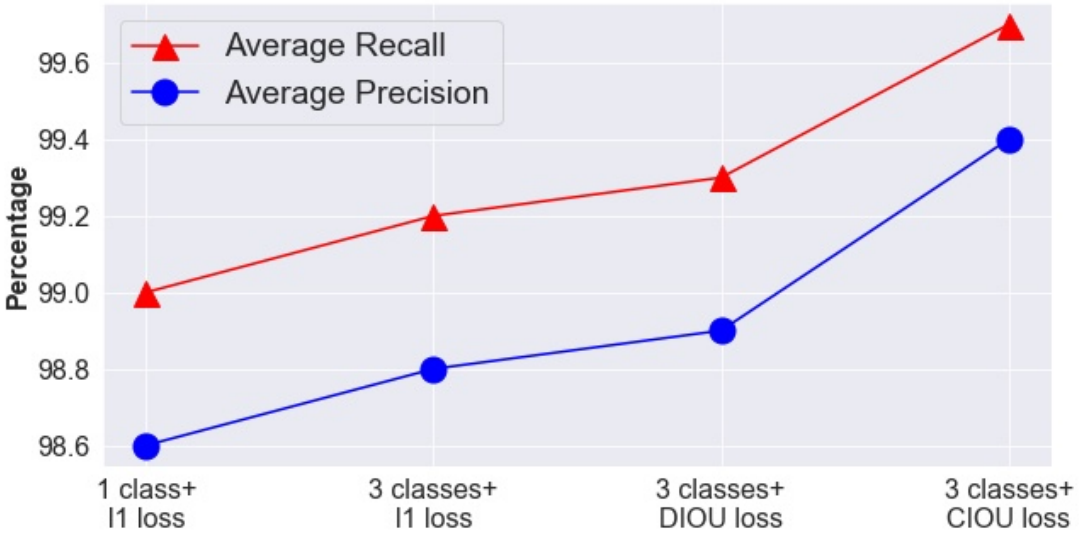


Figure 4.6: **Improvements in table detection performance provided by two model enhancements: Number of classes and regression loss**

Furthermore, we analyzed the impact of our proposed training enhancements; Table Aware Cutout augmentation and dual-step transfer learning. As shown in Table 4.4 we evaluate the impact of multiple augmentation strategies and learning processes on the table detection performance of a model. Good augmentation strategies help object detection models train more efficiently and prevent them from overfitting on the training data. The Detectron2 framework recommends scale jitter [125] as an augmentation strategy to prevent the model from overfitting. Scale Jitter resizes the image at every instance when it is shown to the image with $p = 0.5$ while maintaining the aspect ratio of the image. Furthermore, it does not add or remove any additional data from the image but rather provides different sizes of the objects in the image as it resizes it. However, scale jitter is an efficient strategy for natural images. As this work deals with table detection in datasheet images, we introduce Table Aware Cutout (TAC) strategy which is tailored to table detection.

We compare our proposed augmentation strategy with scale Jitter, Random Erase [128] and Random occlusion [129] as shown in Table 4.4. To test each augmentation strategy we utilized a separate Cascade R-CNN architectures that detects three classes of objects (table, text and figures) to effectively compare their individual impacts. We also compared two learning processes; single-step and dual-step. The table detection performances of the

models are measured in terms of Average Recall (AR) and Average Precision (AP). The models are trained on the training split of the public dataset from Table 4.1. The table detection performances of each model are evaluated on the complete ICDAR 2013 dataset. It is worth noting that no image from this dataset is used to train any of the models.

Table 4.4: **Table detection results comparison between different augmentation strategies and transfer learning processes on the ICDAR 2013 set (complete set) as a test set**

System	Augmentation	Learning Process	Table Detection			Peaked at Epoch	Training Epochs
			Average Recall	Average Precision	Average F1		
Cascade R-CNN	Scale Jitter (Baseline)	Single-Step	99.2	98.8	99	16	50
	Random Occlusion		99.3	99	99	39	50
	Random Erase		99.2	99.1	99	36	50
	Table Aware Cutout		**99.3**	**99.2**	**99.2**	32	50
	Scale Jitter	Dual-Step	99.2	98.8	99	16	20
	Table Aware Cutout		**99.4**	**99.2**	**99.3**	12	+ 20
	Scale Jitter	Dual-Step	99.2	98.8	99	16	20
	Random Occlusion		99.3	99.1	99.2	19	+ 20
	Scale Jitter	Dual-Step	99.2	98.8	99	16	20
	Random Erase		99.2	99.1	99.1	17	+ 20

The dual-step transfer learning process consists of training the model in two steps. In step 1 the model is trained for 20 epochs with scale jitter augmented images so that it could learn about the tables, text and figure objects in the image. As described in Section 4.1.3 and Section 4.3 the model that achieves the highest AP and AR in step 1 is then fine-tuned with TAC augmented images for 20 epochs in step 2. Thus, the dual-step learning process requires that the model should be trained for a total of 40 epochs (20 epochs for step 1 and 20 epochs for step 2). We indicate the sequential number of epochs in this dual-step process in Table 4.4 with a +20 to denote step 2 of the process. On the other hand for the single-step process we trained the models for 50 epochs as denoted in the training epochs column of Table 4.4. Furthermore, as shown in Table 4.4 in the Peaked at Epoch column we also display the epochs at which each augmentation strategy helps the model achieve its highest result. For instance as shown in the column at the 16th epoch with Scale Jitter we get the highest AR and AP with single step process and dual-step process. This model can be use to fine-tune it for table detection in the dual step process for an additional 20 epochs with TAC augmentation where it achieves the highest AR and AP within 12 epochs.

In the single-step process the models are trained for 50 epochs with different augmentation strategies as shown in Table 4.4. TAC augmentation achieves the highest performance

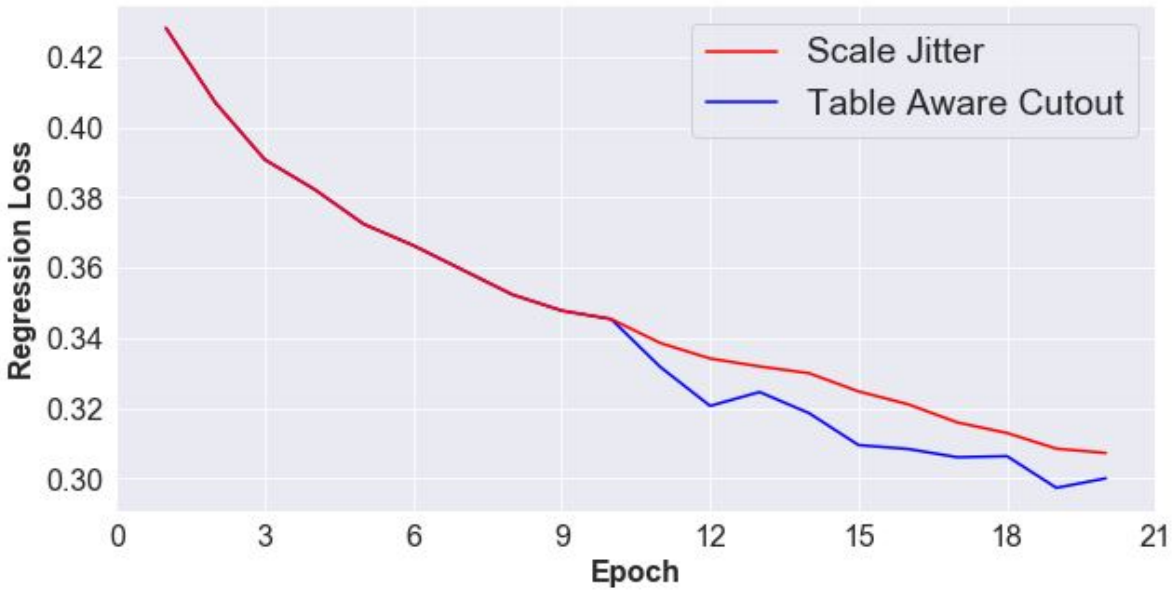


Figure 4.7: **Impact on average regression loss when model is trained using scale jitter and fine-tuned with Table Aware Cutout augmented images**

in terms of AR and AP when compared to other augmentation strategies for this single-step process. Thus, we determined that training the model for a longer period with TAC augmented images could improve the table detection performance of a model. To further reduce the training period, we proposed a dual-step transfer learning process that incorporates TAC augmented images in a transfer learning process to efficiently train a model to address the table detection task. In addition to that, dual-step process also helps the model transfer the object detection knowledge that it has gained from detecting natural objects to detecting page objects such as tables, text and figures in step 1 of the process. As shown in Table 4.4, irrespective of whether the model is trained in a single-step or dual-step process with scale jitter augmented images we achieve the same AR and AP results.

However, when the model is trained with TAC augmented images in a single-step process the AR and AP results improved by 0.1% and 0.4% over the model trained in a single-step process with scale jitter augmented images. Furthermore, in the dual-step process the AR results when TAC augmented images are used further improved by 0.1% as compared to its single-step counterpart. The dual-step process has also shown to improve or provide similar results as the single step process for Random Occlusion and Random Erase in lesser number of epochs. Thus we determined that the dual-step transfer learning process can not only improve the table detection performance of the model but also do so with lesser number of training epochs than a single-step process. In addition to that we show that the model can generalize better to unseen data when fine-tuned with TAC augmented images as no image from ICDAR 2013 dataset is used to train the models.

We calculate and display the average regression loss $l1$ loss of the three box heads at each epoch during the training process of the model in Fig. 4.7. As per our dual-step learning process the model at epoch 10 is fine-tuned with TAC augmented images. As shown in Fig. 4.7 the loss reduces more quickly when the trained model is fine-tuned with TAC augmented images than with scale jitter augmented image. Thus, we show that TAC prevents the model from overfitting on the training data. In addition to that, TAC augmented images also helps the model converge faster as shown in Fig. 4.7.

4.4.2 Final Results

Here we evaluate the performance of the proposed TableDet models to address the table detection task in datasheet images. This work fundamentally deals with improving the table detection performance of the models. This entails accurately locating the object and classifying the object as a table. As shown in Section 4.4.1, the table detection performance

of the models are improved when they trained to detect tables, text and figures in the images. Detecting these three types of objects helps the model distinguish the features between the three object types. Furthermore, we also shown that the table detection performance can be enhanced by utilizing the proposed dual-step transfer learning process.

As per this proposed process the models are trained with Table Aware Cutout augmented images to further enhance their table detection capability while also detecting figures and text in the images. We also proposed utilizing a deformable backbone as a feature extractor to improve this performance. Thus, to analyze the impact of using a deformable backbone over a conventional convolution backbone we propose two model variants of TableDet. We refer to the model with deformable convolution backbone as Model A and the model with conventional convolution backbone as Model B. Both models classify an object as one of three classes: table, figure and text. Except for the different backbones, both models use Complete IOU loss (CIOU) to refine bounding box predictions and are trained as per the dual-step transfer learning process. In a single inference both models would detect tables and classify an entire images into one of two classes. We describe the image classification process in more detail in Section 5.1. On an a single quad core CPU with 64GB RAM system Model A can solve both tasks on a single image in 2.1 seconds.

We used four test sets in this analysis; ICDAR 2013 (complete set), ICDAR 2017 (test set), ICDAR 2019 (test set) and Lytica (test set). Due to the two step incremental nature of the dual-step process we could fine-tune the trained models from step 1 with Table Aware Cutout (TAC) augmented images in step 2 with the individual training sets of three of these datasets. Except for the ICDAR 2013 test set where we use the complete dataset for testing, the remaining three test sets had their respective training set which we could utilize to train the models as per step 2 with TAC augmented images in order to prevent the models from overfitting on the training data.

To fairly compare our table detection performances with state-of-the-art solutions we train our TableDet model variants on the training set of the two combined datasets that we display in Table 4.1. We train the models as per the dual-step transfer learning process. Furthermore, state-of-the-art solutions for ICDAR 2013 (complete set), ICDAR 2017 (test set) and ICDAR 2019 (test set) utilize similar training sets as our public training set from Table 4.1 which has 4200 training images. On the other hand the state-of-the-art solution for the Lytica (test set) utilizes a similar training set as our private training set from Table 4.1 which has 5200 training images.

Thus in step 1 we trained a copy of Model A and Model B on the public training set and another copy of Model A and Model B on the private training set as shown in Table

4.1 with 4200 and 5200 training images respectively. Under the public test sets such as ICDAR 2013 (complete set), ICDAR 2017 (test set) and ICDAR 2019 (test set) we have ensured to use the exact same test split as they were used by the state-of-the-art solutions for these test sets. Similarly we took precautions to ensure that the private Lytica (test set) contains exactly the same 240 test images as the state-of-the-art solution on this set. Furthermore, as we maintained the same test set and a similar training set for each of these test cases we utilize the values reported by the authors of these state-of-the-art solutions.

During the dual-step learning process we measure the performances of the models in both steps in terms of Average Recall (AR) and Average Precision (AP). The models that achieve the highest AR and AP are then compared with state-of-the-art table detection solutions in terms of Recall, Precision and F1-score metrics. Previous state-of-the-art table detection solutions [14, 85, 92] have utilized these metrics at specific IOU values depending on the test set to fairly compare their table detection performances. For the ICDAR 2013 test set, previous table detection solutions have utilized the evaluation metric proposed in [59] to compare their performances [92]. Traquair et al. [12] utilized the same evaluation metric to quantify their table detection performance, under the private Lytica test set.

The state-of-the-art solution as well as most of the previous works under the ICDAR 2017 test set have utilized the evaluation metric proposed in the POD competition [79] to compare their performances to each other [92]. The performances are evaluated at 60% and 80% IOU values. On the other hand, the previous works on the ICDAR 2019 test set have utilized the ICDAR 2019 Table Detection and Recognition competition [8] evaluation metric to measure the improvements provided by their presented solutions [14]. This evaluation criteria involves calculating the F1 scores at IOUs of $60\%, 70\%, 80\%$ and 90% to develop a weighted average F1 score.

Thus, to fairly evaluate our proposed solution with the state-of-the-art solutions we have employed the evaluation metric that is utilized by the state-of-the-art solution for that test set. We have utilized the evaluation metric presented by [59] to measure the improvements provided by our proposed solution on the ICDAR 2013 and Lytica test sets. We have utilized the evaluation metric proposed in [79] and [8] to evaluate our work on the ICDAR 2017 test set and ICDAR 2019 test set respectively. It is worth noting that each of the presented table detection results are averages of three runs. Samples of TableDet detection results are shown in Fig. A.1, Fig. A.2, Fig. A.3, Fig. A.4, Fig. A.5 and Fig. A.6,

The public and private datasets used in this work are created by by combining multiple datasets as shown in Table 4.1. To analyze the table detection performance of our proposed system on the ICDAR 2013 dataset we trained the TableDet model variants on the training

Table 4.5: **Table Detection results comparison with state-of-the-art systems on ICDAR 2013 (complete set) as a test set**

Training set	Test set	System	Recall	Precision	F1
Public dataset (training set)[1]	ICDAR 2013 (complete set)	**Ours (Model A)**	**100**	**100**	**100**
		Ours (Model B)	**100**	**100**	**100**
		CascadeTabNet [14]	100	100	100
		DeCNT [85]	99.6	99.6	99.6
		Huang et al. [92]	100	94.9	97.3
		DeepDeSRT [50]	96.2	97.4	96.8
		TableNet [15]	96.3	96.97	96.6

[1] Public dataset (training set) : Marmot, ICDAR 2017 (training set), ICDAR 2019 (training set)

split of the public dataset. The models are trained as per the dual-step transfer learning process and the performances at are evaluated on the complete ICDAR 2013 dataset. The public dataset comprised of images from the training splits from ICDAR 2017 and ICDAR 2019. It also comprised of the entire Marmot dataset. It is worth noting that no images from ICDAR 2013 dataset are used to train the models. Unlike [14] and [15] we did not use a single image from this dataset to fine-tune either of models. As shown in Table 4.5, both Model A and Model B achieve a 100% F1 score on detecting all 156 tables in this dataset. Thus we can show that the proposed model enhancements, dual-step learning process and augmentation policy together can help the model generalize to unseen data efficiently.

Unlike ICDAR 2017 and ICDAR 2019 datasets, ICDAR 2013 dataset did not have an official training set. Hence in step 2 of the dual-step learning process we fine-tuned Model A and Model B with the entire public dataset. However, the step wise nature of the dual-step learning process provides an option to skip the step 1 of the learning process and directly perform step 2 for these two datasets. For step 1 of the learning process we used the Model A and Model B that achieves a 100% F1 score on the complete ICDAR 2013 dataset. One copy of these models are then fine-tuned on the ICDAR 2017 training set and another copy is fine-tuned on the ICDAR 2019 training set as per step 2 of the learning process. At this step the copies of the models are fine-tuned on their individual training sets with TAC augmented images. Thus, using dual-step transfer learning we could reuse the models that are trained on the public dataset and fine-tune them on their individual training sets. We can thus reduce the training time without compromising on the table detection performance to create models that not only detect tables but also text

and figures.

Table 4.6: **Table Detection results comparison with state-of-the-art systems on ICDAR 2017 test set**

Training set	Test set	System	IOU	Recall	Precision	F1
Public dataset (training set)[1] + ICDAR 2017 (training set)[2]	ICDAR 2017 (test set)	**Ours (Model A)**	60	**99.7**	**98.8**	**99.3**
		Ours (Model B)		99.4	98.5	98.9
		Huang et al. [92]		97.8	97.2	97.5
		DeCNT [85]		97.1	96.5	96.8
		Li et al. [60]		94.4	94.4	94.4
		Ours (Model A)	80	**98.4**	**97.4**	**97.9**
		Ours (Model B)		98.1	97.4	97.7
		Huang et al. [92]		97.5	96.8	97.1
		DeCNT [85]		93.7	96.7	95.2
		Li et al. [60]		90.3	90.3	90.3

[1] Public dataset (training set) : Marmot, ICDAR 2017 (training set), ICDAR 2019 (training set)
[2] Fine tuning dataset

As per the dual step process, at step 2 we fine-tuned the model variants with TAC augmented images from the ICDAR 2017 training set. At each epoch we evaluated the table detection performance of the models on the ICDAR 2017 test set. As shown in Table 4.6, we compared the performances of Model A and Model B against two state-of-the-art table detection solutions with the evaluation metric proposed by Gao et al [79] on this test set. Model A achieves the highest scores on all three metrics and for both IOU values. At 60% IOU it achieves the highest F1 score with 99.3% as compared with [92]. At 80% IOU it achieves a 0.9% and 0.6% increase in recall and precision respectively over [92]. Thus we can show that the deformable variant of TableDet when trained with TAC augmented images in a dual-step learning process can achieve a higher table detection performance than state-of-the-art solutions on ICDAR 2017 test set.

Similar to the fine-tuning process of the Model A and Model B that are trained on ICDAR 2017 training set we also trained a copy of these TableDet variants on ICDAR 2019 training set. Furthermore, the models are trained with TAC augmented images from this training set and their performance is evaluated at each epoch on the ICDAR 2019 test set. We picked the models that achevied the highest AR and AP results in step 2 of the dual-step learning process and measured their performance as per the weighted average F1-score metric set by Gao et al [8]. In Table 4.7 we then compared these results with state-of-the-art table detection solutions. Both Model A and Model B achieve a higher weighted

Table 4.7: **Table Detection results comparison with state-of-the-art systems on ICDAR 2019 test set**

Training set	Test set	System	Intersection over Union (IoU)				Weighted Avg F1
			60	70	80	90	
Public dataset (training set)[1] + ICDAR 2019 (training set)[2]	ICDAR 2019 (test set)	**Ours (Model A)**	96.9	**96.9**	**95.7**	**91.9**	**95.1**
		Ours (Model B)	97.2	96.3	94.9	91.6	94.7
		TableRadar	96.9	95.7	95.1	89.7	94
		NLPR-PAL	**97.9**	96.6	93.9	85	92.7
		CascadeTabNet [14]	94.3	93.4	92.5	90.1	90.1

[1] Public dataset (training set) : Marmot, ICDAR 2017 (training set), ICDAR 2019 (training set)
[2] Fine tuning dataset

average F1 score than the state-of-the-art solutions. Model A achieves the highest F1 score score on all IOU values except for 60% as compared to the other systems. It is worth noting that both Model A and Model B achieve higher F1 scores at 90% IOU value as compared to the other systems. This indicates that both models can accurately detect tables at high IOU values.

TableRadar system which is the top table detector in the ICDAR 2019 competition. To display the performance improvements brought on by our proposed model and training enhancements we show the F1 scores achieved by our two model variants and TableRadar on Fig. 4.8. Model A achieves a highest F1 score at all IOU values except at 60% where it shares the same score as TableRadar. We can thus show that the proposed TableDet models when trained with dual-step learning process not only reduce the training time but can also detects tables at multiple IOU values efficiently.

Table 4.8: **Table Detection results comparison with state-of-the-art systems on Lytica test set**

Training set	Test set	System	Recall	Precision	F1
Private dataset (training set)[1]+ Lytica (training set)[2]	Lytica (test set)	**Ours (Model A)**	**99.5**	**99.9**	**99.7**
		Ours (Model B)	**99.5**	97.7	98.6
		Traquair et al. [12]	95.3	91.1	93.2

[1] Private dataset (training set) : Marmot, ICDAR 2017 (training set), ICDAR 2019 (training set), Lytica (training set)
[2] Fine tuning dataset

As shown in Table 4.1 the private dataset comprises of the public dataset and the Lytica

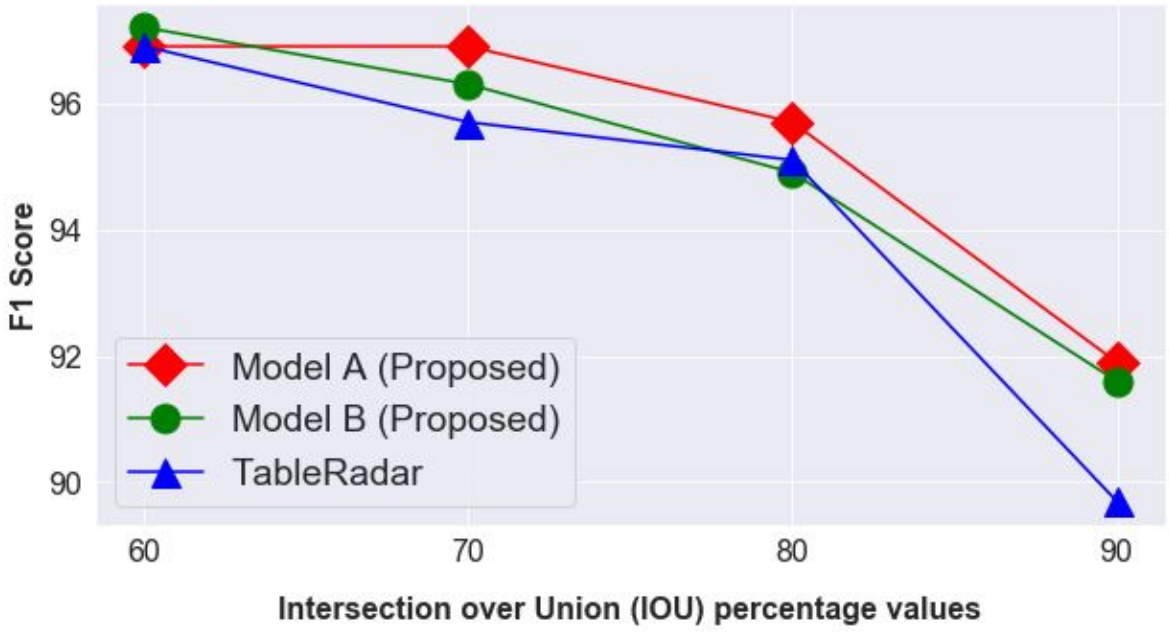


Figure 4.8: **Table detection performance on ICDAR 2019 (test set) of Model A and Model B in terms of F1 scores on four IOU values against state-of-the-art system, TableRadar**

dataset. When we trained the TableDet model variants for ICDAR 2017 ad ICDAR 2019 datasets we fine-tuned the models with the respective training sets in step 2 of the learning process. We followed a similar approach to training the models for the Lytica test set. Both Model A and Model B are trained on the private dataset for step 1 of the process. Then the models are fine-tuned in step 2 with TAC augmented images from the Lytica training set. This allowed us to fairly compare our table detection performances with other state-of-the-art solutions such as [12] that are trained on a similar private dataset. As shown in Table 4.8 we achieve a high score on all three metrics against [12]. Model A once again achieves the highest F1 score at 99.7% while Model B is only marginally lower with 98.6%.

It is also worth noting that the public and private datasets used in this work comprise not only datasheet images but also images of scientific papers and reports. Furthermore, the tables in all datasets appeared with different scales and orientations. While TableDet is developed to accurately detect tables in datasheet images, the table detection scores achieved on all four test sets indicate that it can be generalized to accurately detect tables in other types of document images as well. Thus, we can show that our proposed enhancements improves the model's table detection performance and can help it detect tables at multiple scales and orientations efficiently.

4.5 Conclusion

In this chapter we propose TableDet, a deep learning based methodology that can address the first stage of the table text extraction pipeline. It can address both table detection and table image classification in datasheet (document) images in a single inference. We show that detecting tables, text and figures enables TableDet model to enhance its table detection performance. TableDet utilizes a Cascade R-CNN architecture with Complete IOU (CIOU) loss at each box head and a deformable convolution backbone to capture the variations in scale and orientations of tables and to detect them accurately. In addition our proposed dual-step transfer learning process can efficiently develop a single model to classify each object in an image as a table, text or figure. We also present the impact of using Table Aware Cutout (TAC) augmented images in this training process to fine-tune and to improve the model's table detection performance.

Furthermore, utilizing TAC augmented images can help the models generalize better to unseen data and prevent it from overfitting on the training data. These enhancements make TableDet Model A which has a deformable backbone achieve the highest table detection performance on all four test sets. TableDet Model A achieves 100% F1 score on the ICDAR

2013 (complete set) when no image from this set is used to train or fine-tune the model. On the ICDAR 2017 (test set) and Lytica (test set) Model A achieves the highest F1 score as compared to state-of-the-art table detection solutions. On the ICDAR 2019 (test set) it achieves the highest weighted average F1 score with 95.1% which indicates that it can detect tables at higher IOU values. Thus, we present TableDet a table detection solution that improves table detection performance on three public and one closed dataset.

Our future agenda for this task includes classifying the tables based on the presence of line separators so that we can further enhance the quality of table structure detection solutions. In addition we would also like to improve the application of TAC augmented images to reduce the training of the TableDet model from a dual step process to a single step process. Our future agenda also includes enhancing the deformable convolution operation of the TableDet model to only consider only a limited number of table orientations to help the model to pay attention to other complex features of the table.

Chapter 5

Table Image Classification

Documents and datasheets contain multiple pages about various topics such as scientific reports [79, 8] or product specifications or administrative reports by governments [68]. On each page there are multiple type of objects or components displayed such as tables, paragraphs of text which we refer to as text and figures which include charts, maps, diagrams and icons. There have been previous works which have dealt with identifying only table objects such as [14, 12, 13]. There have a been a some works that have detected other objects on the image along with tables such as [141, 6] in images and [142] in PDF documents. Such table detection methods have not considered possibility of identifying the images that do not contain tables. As shown in Fig. 1.1 in our presented table text extraction pipeline we propose eliminating images without tables at stage 1 so that such images can be avoided from being processed by the later stages. By identifying such images, we hypothesize that only images with tables can be sent for further analysis. The analysis in the later stages of the pipeline includes identifying the components of the detected tables which can be decomposed into rows, columns or cells and extracting the text from these components. Up until this work, there has been no work conducted in classifying such full page images into two classes; images with tables (Table image) and images without tables (Non-Table images).

In this thesis, we propose a table text extraction pipeline with three stages as shown in Fig. 1.1. Stage 1 deals with table detection and table image classification. We describe the steps taken by us to develop the proposed TableDet model which include the training enhancements in Section 4.1. We further describe the performance improvements it achieves in detecting tables on multiple public and one closed test set over state-of-the-art solutions in Section 4.4. Thus, we propose utilizing the enhanced table detection performance of this TableDet model to solve an additional task in stage 1 of the pipeline; table image

classification. As the model can detect figures and text objects along with tables in all datasheet (document) images we propose utilizing the object detection results to classify the entire image into two classes.

We describe the steps taken by us to classify the images as Table and Non-Table images in Section 5.1. We provide details on the public and closed test sets used in this study in Section 5.2. The performance improvements provided by the proposed solution are provided in the Section 5.4. Finally we provide a brief summary of the enhancements and performance improvements brought on by out proposed solution in Section 5.5.

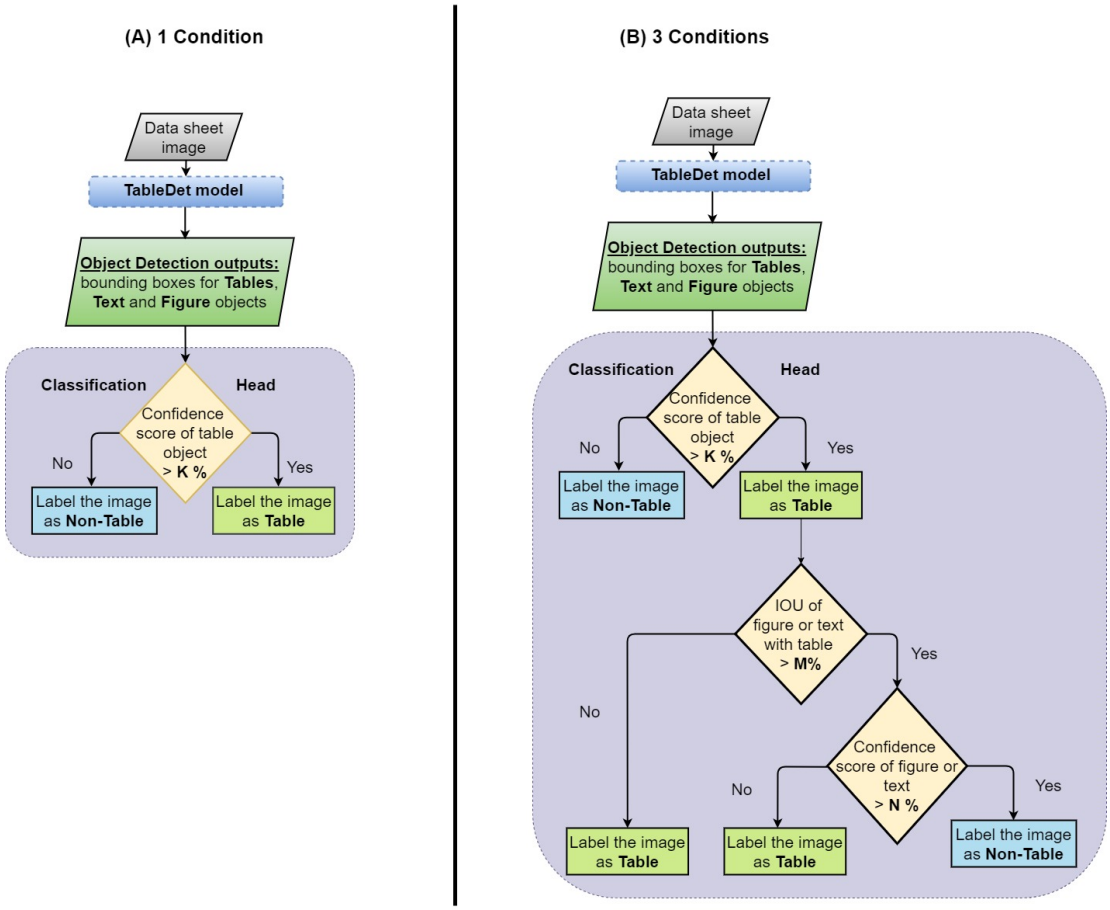


Figure 5.1: **Enhancements to improve table image classification performance of TableDet**

5.1 System Model

Deep learning based methods have excelled in object detection as well as on image classification on multiple large scale datasets such as COCO [114] and ImageNet [138]. Furthermore, such methods can be utilized on datasets that do not contain naturally occurring objects such as medical images [101, 117] or geographical images [47, 48]. Despite the lower number of images in such non-naturally occurring image datasets, deep learning based methods can excel in classifying and detecting objects in such images due to the concept of transfer learning. This learning process entails training deep learning architecture on a large dataset such as COCO to classify images or detect objects in images and then using the image and object feature knowledge it has gained to train it on another dataset that comprises of completely different images. This process of transfer learning can be applied to image classification tasks as well as object detection tasks [47, 48, 14]. Furthermore, these architectures also present their generalization capability as they achieve higher the image classification and object detection performances on these new datasets. Thus, due to its generalization potential along with its feature of transfer learning we propose a deep learning based approach to address the task of table image classification in datasheet (document) images.

Thus, we propose the TableDet model a deep learning based approach as shown in Fig. 4.3 to address table detection and table image classification in a single inference in datasheet (document) images. In Section 4.1 we describe the steps we take to develop this approach to accurately detect tables in datasheet (document) images. We employ a Cascade R-CNN architecture [11] to build the TableDet model. In addition to that we utilize a deformable convolution [55] backbone and utilize Complete IOU loss [18] to improve its table detection performance. We also detect figures and text objects along with tables to help the model distinguish between these three types of objects. Furthermore, the model is trained in a dual-step transfer learning process with our proposed Table Aware Cutout augmentation strategy to improve its table detection performance. As shown in Fig. 4.3 in addition to using the three successive box heads of the Cascade R-CNN architecture we introduce a classification head that is used only during inference time and does not require any training.

As shown in Fig. 4.2 we do not train the classification head of the TableDet model. Instead throughout the dual-step transfer learning process the model is trained to only improve its table detection performance while also detecting text and figure objects in the images as shown in Fig. 4.2 (A). During the inference process we introduce a classification head that accepts the detection predictions from the final box head as an input as shown in Fig. 4.2 (B). These detection predictions include the bounding boxes with their respective object class labels and confidence scores which are considered as object detection outputs

as shown in Fig. 4.3. The object class labels denote the predicted class of the detected object. Since the TableDet model detects only tables, text and figures, the predicted class has to be one of these classes. As shown in Fig. 4.2 (B) the classification head evaluates the object class label and the confidence score of the detected objects. If it finds a table object among the object labels with a confidence score that is greater than $K\%$ then it predicts the image label as Table else it predicts the image label as Non-Table.

As shown in Fig. 5.1 in addition to the single condition to determine the image label we add two more conditions that will further enhance the image classification performance of the TableDet model. After the first condition if the image is labeled as Non-Table the classification head does not perform any further steps. If the image is labeled as Table then as a second condition the classification head measures if there are any figures or text objects with greater than $M\%$ IOU value over the detected table object bounding box. In an instance where no such overlapping object appears over the detected table then the image is labeled as Table. However, if there is such an object then the third condition will evaluate if the confidence score for this overlapping object is lower than $N\%$. We determined that figure or text objects have a lower confidence score when they overlap with table objects with greater than $M\%$ IOU value. Thus, if the overlapping object has a confidence score of lower than $N\%$ then the image is labeled as Table else it is labeled as Non-Table. Thus, the classification head labels the image as a Table if it detects a table object with greater than $K\%$ confidence score and if there are no overlapping objects detected in the same region as the table. Additionally if there is a figure or text object that overlaps the detected table by more than $M\%$ IOU value then the classification head can label the image as a Table the overlapping object has a confidence score lower than $N\%$. On the other hand it can label an object as Non-Table if it does not detect a table object.

We determined that these enhancements can help the TableDet model identify all Table images while also improving its classification potential for Non-Table images. With the addition of the classification head the TableDet model can solve the table image classification task by predicting the labels of images based on a set of three conditions as shown in Fig. 4.2 and Fig. 5.1. In addition to the enhancements that enable TableDet to address the table image classification task it also solves this task without any additional training. Furthermore, during the inference process TableDet produces object detection outputs (bounding boxes with confidence scores and object class labels) and image classification output (image labels) as shown in Fig. 4.3. These outputs are provided for each image as a combined output. Thus, a TableDet model can detect tables as well as classify the entire image in a single inference.

As shown in Fig. 5.2, as a post-processing step, images that are predicted as Table

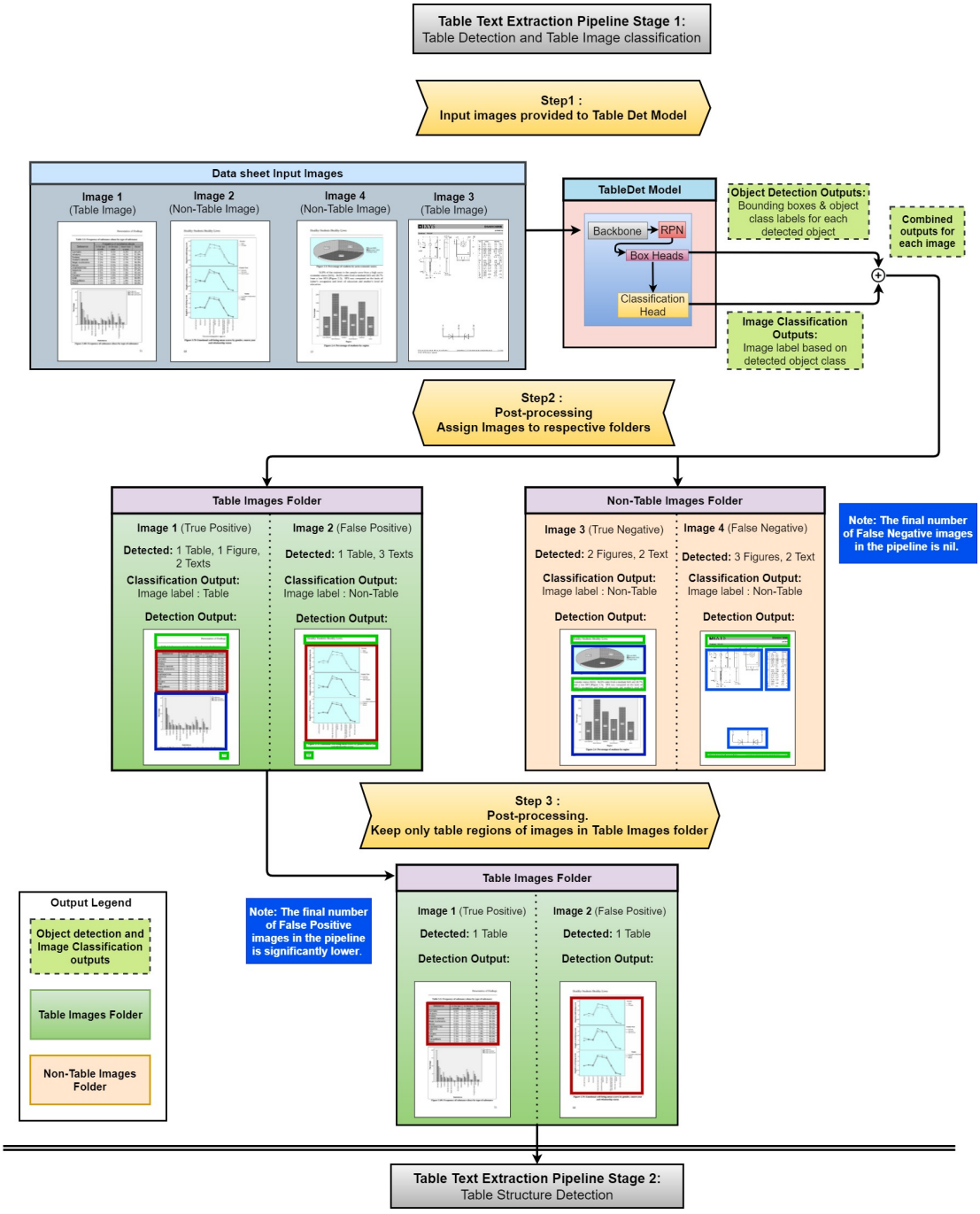


Figure 5.2: **Proposed TableDet approach to detect tables, figures and text in an image and then classify the images**

are assigned to the Table images folder. This folder will contain images that are correctly predicted as Table (True Positive) and those that are incorrectly predicted as Table but are actually Non-Table images (False Positive). Similarly, images that are predicted as Non-Table are assigned to the Non-Table images folder which contains images that are correctly predicted as Non-Table (True Negative) and those that are incorrectly predicted as Non-Table but are actually Table images (False Negative). For visualization purposes, we have shown some possible False Positive and False Negative images in Fig. 5.2.

The TableDet models is enhanced to accurately detect tables, while also detecting figures and text. We determined, that due to the enhanced table detection capability of the TableDet model along with the enhancements on the classification head, the number of incorrectly classified Non-Table (False Positive) and Table (False Negative) images in the table text extraction pipeline are very low. We determined that the final number of False Positive images in the pipeline can be significantly reduced while the final number of False Negative images can be close to nil. As an additional post-processing step as shown in Fig. 5.2, only the images in the Table images folder with their respective table detections (locations) are sent to the next stage of the table text extraction pipeline. By classifying the images in this stage of the table text extraction pipeline we can prevent the next stage from dealing with Non-Table images. Thus the immediate next stage which deals with table structure detection [56, 17, 16] would save some processing time.

5.2 Datasets

In this work, we have utilized four public and one closed document table detection dataset to train and test the proposed Model A and Model B variants of the TableDet model for table detection task. As the classification head component of the TableDet model is used only at inference time, we do not train it on these datasets for table image classification. We do however train standard image classification models for table image classification to compare their performance with the TableDet model variants. As shown in Table 4.1 and described in Section 4.3 we train and test two copies of the TableDet model variants on the public and private datasets individually for the table detection task. To fairly compare the TableDet model with the standard image classifiers we have maintained the same train and test splits of the public and private datasets as shown in Table 5.1.

The public available dataset consists of images from ICDAR 2013 [68], ICDAR 2017 [79], ICDAR 2019 [8] and Marmot [1] as shown in Table 5.1. The closed dataset comprises of

[1] http://www.icst.pku.edu.cn/cpdp/sjzy/index.htm

a commercial dataset which we refer to as the Lytica dataset and the public dataset images as shown in Table 5.1. We describe each individual datasets in more detail in Section A.2.1. Furthermore, we have labeled the images as Table or Non-Table depending on whether a table object is present in the image or not. For instance if there are table, text and figure objects in the image then we labeled the image as Table. On the other hand if there is only text and figures object in the image we labeled it as a Non-Table image.

Table 5.1: **Datasets used to train and test TableDet**

Set name	**Datasets**	**Train**			**Test**		
		Table	**Non-Table**	**Total**	**Table**	**Non-Table**	**Total**
Public dataset	ICDAR 2013	-	-	-	128	110	238
	ICDAR 2017	557	1043	1600	244	573	817
	ICDAR 2019	600	-	600	-	-	-
	Marmot	1006	994	2000	-	-	-
	Total	2163	2037	4200	372	683	1055
Private dataset	Lytica	1000	-	1000	126	114	240
	Public dataset	2163	2037	4200	-	-	-
	Total	3163	2037	5200	-	-	240

The ICDAR 2013 dataset includes 238 images. It is worth noting that none of these images were used to train either image classifiers for table image classification nor the TableDet model for table detection as shown in Table 5.1 and Table 4.1 respectively. The ICDAR 2017 dataset includes 1600 and 817 images for training and testing. This test set consists of 244 Table and 573 Non-Table images. It is worth noting that the number of Non-Table images is close to double the number of Table images in both the training and test sets. In this study we utilize the modern document subset of the ICDAR 2019 dataset which comprises of 600 training and 240 test images all of which are Table images. The complete Marmot dataset comprises of 2000 document images. We utilize the entire dataset to train the TableDet models for the table detection task. Furthermore, we have utilized Lytica dataset which comprises of 1000 Table images that are used for training the TableDet model for table detection while the test set consist of 114 Non-Table images along with 126 Table images.

5.3 Training Details

The TableDet model variants (Model A and Model B) are trained to solve the table detection task with the dual-step transfer learning process. As per the process, a model is trained at step 1 for 20 epochs with scale jitter augmented images to detect tables, text and figures. At each epoch the model performance is tracked in terms of Average Recall (AR) and Average Precision (AP). The model that achieves the highest AR and AP at a particular epoch is then fine-tuned in step 2 with Table Aware Cutout (TAC) augmented images. The step wise nature of the learning process, the Table Aware Cutout augmentation and model enhancements such as detecting three classes of objects enhances the table detection performance of these models.

We address the table image classification task by utilizing the object detection results of a trained TableDet model as shown in Fig. 5.1 and Fig. 4.2. A trained TableDet model will produce object detection outputs such as bounding boxes with their respective object class labels and confidence scores. As described in Section 5.1 during the inference process we attach a classification head to the model to provide image labels as outputs. The classification head determines the image label as Table or Non-Table based on three conditions. As shown in Fig. 5.3 the first condition checks if the detected object is a table and if its confidence score is greater than $K\%$ which we determined should be set at 50%. If the image satisfied this condition then the image would be labeled as Table and would then face the second condition where the classification head measures if there are figure or text objects that overlap with the detected table object. However, if the image does not pass the first condition then the image is labeled as Non-Table.

In the instance the image is labeled as Table after passing the first condition, in the second condition the classification head measures the IOU value of figure and text objects that may overlap with the detected table object. If such a figure or text object with an IOU value greater than $M\%$ exists, then the image would be sent for further processing to the third condition. We determined that the value of $M\%$ should be set at 90% to achieve optimal results. If such an object does not exist then the image is labeled as Table. In the third condition the confidence score of the overlapping object is evaluated to determine the image label. If the object has a confidence score greater than $N\%$ which we determined should be set at 30% as shown in in Fig. 5.3 , then the image is labeled as Non-Table else it is labeled as Table.

Thus, with a set of three conditions the classification head of the TableDet model can provide labels for each image. It is worth noting that the TableDet models can classify images as Table or Non-Table without any additional training for this task. However, to

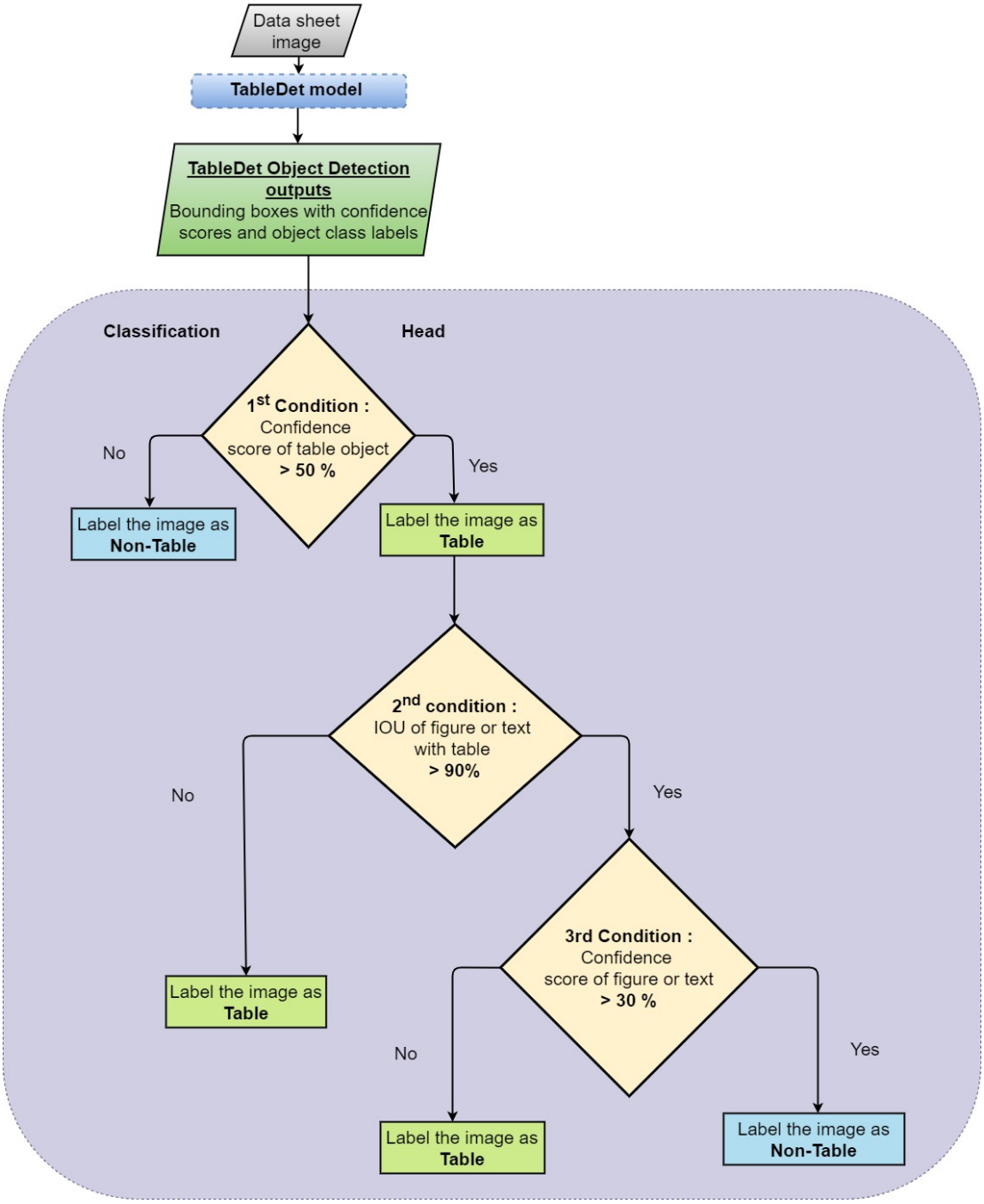


Figure 5.3: **TableDet classification head with three conditions to classify the image as Table or Non-Table**

compare the table image classification results of the TableDet model variants we trained two image classification architectures; ResNet-50 and ResNet-101. We trained both models with a linear learning rate of 0.03 and use scale jitter augmented images to prevent the models from overfitting. The models are trained for 30 epochs on the public and private dataset training splits as shown in Table 5.1.

5.4 Results

Our proposed TableDet model variants (Model A and Model B) are trained to accurately detect tables in datasheet (document) images. TableDet Model A utilizes a deformable convolution backbone while TableDet Model B utilizes a conventional convolution backbone. Both models utilize CIOU bounding box regression loss and have are trained with the dual-step transfer learning process and TAC augmented images. We also propose utilizing the enhanced table detection capability of the TableDet model variants to address the table image classification task as proposed by our table text extraction pipeline. As shown in Fig. 4.3 we train the TableDet model to produce object detection outputs. These outputs are the bounding boxes to denote the location of the detected table text or figure objects along with the respective confidence scores and object class labels. At inference time a classification head is attached to the model as shown in Fig. 4.3 and Fig. 5.1. It does not require any additional training instead it uses the object detection labels and confidence scores to predict the image label.

As shown in Fig. 5.1 as per the first condition if a table object is detected in the image with greater than 50% confidence score then the image is labeled as Table else it is labeled as Non-Table. If the image is labeled as Table by the first condition then the classification head evaluates the image under two additional conditions to improve the overall table image classification performance. The second condition checks if the table object overlaps with a figure or text object. This condition checks if the IOU value between the table and the figure or text objects is greater than 90%. If there is no such overlapping object then the image is labeled as Table. However, if there is a figure or text object that overlaps the table with greater than 90% IOU then the image is evaluated under the third condition. At this condition the confidence score of the figure or text object is measured and if it is less than 30% then the image is labeled as Table else it is labeled as Non-Table. The final results of the TableDet model at inference time are the object detection outputs as well as an image label (image classification output) as shown in Fig. 4.3. Thus, a TableDet model can address both table detection and table image classification in a single inference. Furthermore, as a post-processing step, the images are assigned to Table images folder and

Non-Table images folder as shown in Fig. 5.2.

To compare the table image classification performances of the TableDet model variants we train two image classification architectures; ResNet-50 and ResNet-101 as shown in Table 5.2 and Table 5.4. We use ICDAR 2013 (complete set), ICDAR 2017 (test set) and Lytica (test set) to perform this evaluation. The TableDet models (Model A and Model B) have been trained as per the dual-step transfer learning process on the public and private dataset for the table detection task as shown in Table 4.1. It is worth noting that the TableDet models are not trained for the table image classification task instead the models utilize their enhanced table detection performance to determine the label for the image. Except for models tested on the ICDAR 2013 test set all the other models are fine-tuned on their individual training sets in step 2 of the process. The image classifiers have also been trained on the public and private dataset and fine-tuned with individual training sets except for the case of ICDAR 2013 test set. While the TableDet models are not trained for this task, the image classifiers are trained to classify each image as Table or Non-Table. Furthermore, we present the impact of utilizing 3 conditions in the classification head of the TableDet models over 1 condition in Table 5.2 and Table 5.4.

In this work we consider the Table images as the positive class and the Non-Table images are considered as the negative class. As a post-processing step the TableDet models assign the images into a folder based on the predicted image label. If the predicted label of the image is Table then the image is assigned to the Table images folder else it is assigned to the Non-Table images folder as shown in Fig. 5.2. We follow the same post-processing step for the image classifiers to fairly compare the performance with the TableDet models. Thus, The Table images folder contains True Positive images which are Table images accurately classified as Tables. It also contains False Positive images which are Non-Table images incorrectly classified as Table images. In the Non-Table images folder we have False Negative and False Positive images. The False Negative images are actual Non-Table images that are correctly identified as Non-Table images. The False Positive images are actual Non-Table images that are the incorrectly classified Table images.

As shown in Table 5.2 we denote the number of True Positive, False Positive, True Negative and False Negative images that are classified by the TableDet model variants (with 1 condition and 3 conditions) and the image classifiers on the three test sets. In addition to that we also display the True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR) and False Negative Rate (FNR) achieved by each solution in Table 5.3. Furthermore, up until this work there has been no previous work in table image classification and hence we propose utilizing Recall, Precision and F1 score to compare the results between the TableDet model variants and the image classifiers.

Table 5.2: **Table image classification performance comparison**

Test Set	System	Conditions	Table Images Folder		Non-Table Images Folder	
			True Positive	**False Positive**	**True Negative**	**False Negative**
ICDAR 2013 (complete set)	ResNet-50 [1]	-	125	5	105	3
	ResNet-101 [1]	-	126	**0**	**110**	2
	Model A	1	128	15	95	0
	Model B	1	128	28	82	0
	Model A	**3**	**128**	1	109	**0**
	Model B	3	128	3	107	0
ICDAR 2017 (test set)	ResNet-50 [1]	-	225	5	568	19
	ResNet-101 [1]	-	226	**3**	**570**	18
	Model A	1	244	41	532	0
	Model B	1	244	54	519	0
	Model A	3	**244**	16	557	**0**
	Model B	3	244	18	555	0
Lytica (test set)	ResNet-50 [2]	-	119	15	99	6
	ResNet-101 [2]	-	121	**13**	**101**	5
	Model A	1	126	15	99	0
	Model B	1	126	25	89	0
	Model A	3	**126**	10	104	**0**
	Model B	3	126	12	102	0

[1] Trained on Public dataset (training set) : Marmot, ICDAR 2017 (training set), ICDAR 2019 (training set)
[2] Trained on Private dataset (training set) : Marmot, ICDAR 2017 (training set), ICDAR 2019 (training set), Lytica (training set)

Table 5.3: **Table image classification performance comparison per rate**

Test Set	System	Conditions	Table Images Folder		Non-Table Images Folder	
			True Positive Rate [%]	**False Positive Rate [%]**	**True Negative Rate [%]**	**False Negative Rate [%]**
ICDAR 2013 (complete set)	ResNet-50 [1]	-	97.7	4.5	95.5	2.3
	ResNet-101 [1]	-	98.4	**0**	**100**	1.6
	Model A	1	**100**	13.6	86.3	**0**
	Model B	1	**100**	25.5	74.5	**0**
	Model A	**3**	**100**	0.9	99.1	**0**
	Model B	3	**100**	2.7	97.3	**0**
ICDAR 2017 (test set)	ResNet-50 [1]	-	92.2	0.9	99	7.8
	ResNet-101 [1]	-	92.6	**0.5**	**99.5**	7.4
	Model A	1	**100**	7.2	92.8	**0**
	Model B	1	**100**	9.4	90.6	**0**
	Model A	3	**100**	2.8	97	**0**
	Model B	3	**100**	3.1	96.9	**0**
Lytica (test set)	ResNet-50 [2]	-	95.2	13.2	86.8	4.8
	ResNet-101 [2]	-	96	11.4	88.6	4
	Model A	1	**100**	13.2	86.8	**0**
	Model B	1	**100**	21.9	78.1	**0**
	Model A	3	**100**	**8.8**	**91.2**	**0**
	Model B	3	**100**	10.5	89.5	**0**

[1] Trained on Public dataset (training set) : Marmot, ICDAR 2017 (training set), ICDAR 2019 (training set)
[2] Trained on Private dataset (training set) : Marmot, ICDAR 2017 (training set), ICDAR 2019 (training set), Lytica (training set)

A higher recall score and TPR indicates that the model can identify most of the Table images. A higher precision score and a higher FPR indicates that there would be a lower number of Non-Table images that are incorrectly classified as Table images in our proposed the table text extraction pipeline. We display the performances of the models in terms of Recall, Precision and F1 score in Table 5.4. It is worth noting that the motivation for this work is to find a solution that correctly identifies all True Positive (Table) images and only minimizes the number of False Positive (Non-Table classified as Table) images fed into the pipeline. This implies that the solution does not produce any False Negatives (Table images classified as Non-Table) and accurately identifies most if not all the False Negative (Non-Table) images.

The ResNet-101 architecture achieves 100% precision score and 100% TNR on the ICDAR 2013 (complete set) which indicates that it identifies all Non-Table images as displayed in Table 5.4. However, both image classification models achieve lower recall scores and FNR values than both TableDet models. All 128 tables of the complete dataset are accurately labeled as Table by both TableDet model variants as shown in Table 5.2 irrespective of the number of conditions utilized in the classification head. Furthermore, Model A with 3 conditions also incorrectly classifies only 1 out of 110 Non-Table images. Thus, it would send only 1 image and all the Table images from this dataset for further processing in the next stage. In addition to that, the results on this test set also present the impact of utilizing 3 conditions over 1 condition in the classification head. By utilizing 3 conditions instead of 1 condition the recall value remains at 100% while the precision value increases by 9.7% and 15.7% for Model A and Model B respectively. This is also indicated by the 12.8% increase in TNR value when 3 conditions are utilized in place of a single condition as shown in Table 5.3. It is also worth noting that none of the ICDAR 2013 dataset images are used to train any of the TableDet models and image classifiers for this task.

The ResNet-101 model once again achieves the highest precision score of 98.7% and TNR of 99.5% as compared to the other systems on the ICDAR 2017 test set. However, TableDet model variants irrespective of the number of conditions in the classification head achieve 100% recall which is validated by the 100% TPR values that each model achieves. In addition to that, as shown in Table 5.4 Model A with 3 conditions in the classification head achieves 93.9% precision. This denotes that 557 images out of the total 573 Non-Table images would be eliminated from the pipeline as shown in Table 5.2. Only 16 Non-Table images along with all 244 Table images would be sent for further processing. Similar to the performance improvement on the ICDAR 2013 complete set, when 3 conditions are used in the classification head the recall value and TPR value stays at 100% while the precision value and TNR improves by more than 8% and 4.2% for both TableDet model

Table 5.4: **Table image classification performance comparison**

Test Set	System	Conditions	Recall	Precision	F1
ICDAR 2013 (complete set)	ResNet-50 [1]	-	97.7	96.2	98.9
	ResNet-101 [1]	-	98.4	**100**	99.2
	Model A	1	100	89.5	94.5
	Model B	1	100	82	90.1
	Model A	3	**100**	99.2	**99.6**
	Model B	3	100	97.7	98.8
ICDAR 2017 (test set)	ResNet-50 [1]	-	92.2	97.8	94.9
	ResNet-101 [1]	-	92.6	**98.7**	95.6
	Model A	1	100	85.6	92.2
	Model B	1	100	81.8	90
	Model A	**3**	**100**	93.9	**96.8**
	Model B	3	100	93.1	96.4
Lytica (test set)	ResNet-50 [2]	-	95.2	88.8	91.9
	ResNet-101 [2]	-	96.0	90.2	93.6
	Model A	1	100	89.4	94.4
	Model B	1	100	83.4	90.9
	Model A	**3**	**100**	**92.6**	**96.2**
	Model B	3	100	91.3	95.5

[1] Trained on Public dataset (training set) : Marmot, ICDAR 2017 (training set), ICDAR 2019 (training set)

[2] Trained on Private dataset (training set) : Marmot, ICDAR 2017 (training set), ICDAR 2019 (training set), Lytica (training set)

variants, respectively. Thus, utilizing 3 conditions only 16 and 18 Non-Table images are incorrectly classified by the TableDet Model A and Model B respectively as compared to using 1 condition where a higher number of Non-Table images are incorrectly classified.

As compared to the other datasets, the Lytica dataset consists of more complex tables. These tables contain multiple merged cells to layout the information in a hierarchical format as shown in Fig. A.7. With the enhanced table detection capability and the introduction of 3 conditions in the classification head, the TableDet Model A is able to achieve 100% recall and 92.6% precision for table image classification task on the Lytica dataset. As shown in Fig. A.9 the Lytica images that were actually Non-Table images but incorrectly classified as Table images by the TableDet Model A. These images contain graphs with line separators to denote the co-ordinates. These graph objects are incorrectly classified as table objects due to the presence of such lines which resemble a line separated table. Thus, the TableDet model achieves a lower precision score than recall score as it is unable to accurately classify such objects as graphs.

Furthermore, as shown in Table 5.2 and Table 5.3 by utilizing 1 or 3 conditions in the classification head of TableDet Model A, it correctly identifies all 126 Table images and achieves 100% TPR. In addition to that, by utilizing 3 conditions instead of 1 condition the number of incorrectly classified Non-Table images drops to 10 from 15. Thus, the TableDet Model A with 3 conditions in the classification head not only identifies all Table images but also further reduces the number of Non-Table images that would have been processed in the successive stages of the table text extraction pipeline.

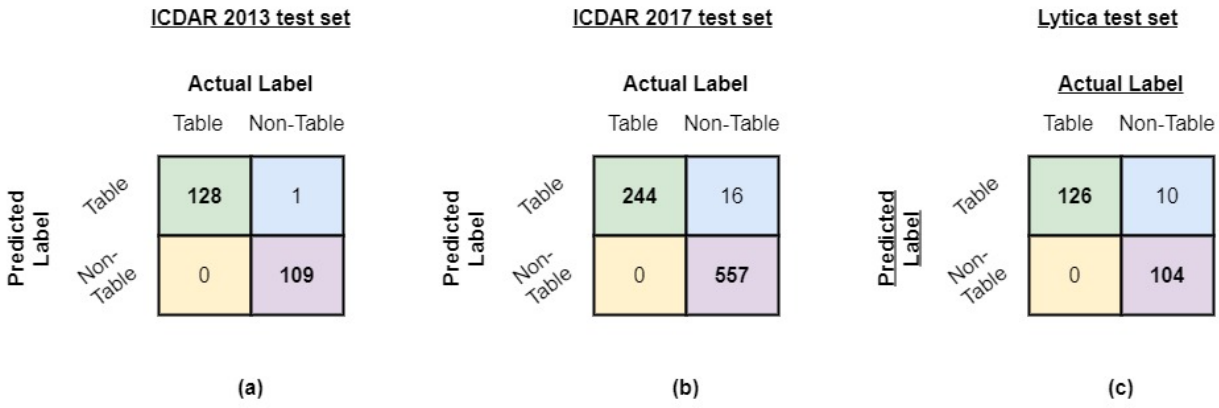


Figure 5.4: **Confusion matrices for inference results on each test set by TableDet Model A with 3 conditions in the classification head**

At this stage of the table text extraction pipeline our motivation is to develop a solution

that identifies all Table images and reduces the number of Non-Table images that would be processed in the next stage of the pipeline. While the ResNet models do achieve a higher precision scores on two test sets, they incorrectly label a few Table images. This denotes that a few Table images would not be sent for further processing to the next stage of the pipeline. On the other hand the TableDet models ensure that all Table images are accurately identified while eliminating a large number of Non-Table images from the proposed pipeline. In addition to that the TableDet models can address the task of table image classification without any additional training.

Furthermore, utilizing 3 conditions in the classification head of the TableDet models enables them to further reduce the number of Non-Table images. The TableDet Model A with its enhanced table detection capability coupled with 3 conditions in the classification head achieves the highest F1 score on all three test sets as shown in Table 5.4. It also consistently achieves the highest TPR value and the lowest FNR value on all three test sets as shown in Table 5.3. As shown in Fig. 5.4 TableDet Model A with 3 conditions not only captures all the Table images but also a significant number of Non-Table images. Furthermore, as indicated in Fig. 5.4 no Table image is incorrectly classified as a Non-Table image by this model. This indicates that it will not avoid any Table images from these three test sets.

Hence, we determined that it would be beneficial to use an object detection model like TableDet Model A with 3 conditions in the classification head instead of an image classification model to not only reduce the number of Non-Table images but to capture all Table images. We display the inference results of TableDet Model A in Fig A.8, Fig. A.9 and Fig. A.10.

5.5 Conclusion

The task of table image classification requires classifying a full page datasheet (document) images as a Table or Non-Table image. The motivation of this task is to reduce the number of images without table objects (Non-Table) in the proposed table text extraction pipeline while ensuring that all images with table objects (Table) are processed. Thus, we propose utilizing the enhanced table detection performance of the TableDet models to address this task without any additional training. The Model A variant of TableDet which has a deformable backbone, CIOU bounding box loss and utilizes 3 conditions in the classification head is able to achieve 100% recall and 100% True Positive Rate on the ICDAR 2013 complete set, ICDAR 2017 test set and Lytica test set. Furthermore, it is able to achieve above 92% precision along with achieving the highest F1 score on all three

test sets. This denotes that it would allow about 8% of Non-Table images to remain in the pipeline.

Chapter 6

Table Structure Detection

Global organizations such as component distributors, manufacturers and product designers share information with each other through various types of documents such as datasheets. The important information in the documents is generally presented in tabular format. Extracting this data and making available for analysis can organizations to take data-driven [5]. We hypothesize that the task of extracting this data can be dealt in three stages as shown in our proposed table text extraction pipeline in Fig. 1.1.

The proposed pipeline functions on datasheets (document) full page images. At stage 1 of the pipeline we propose table detection and table image classification steps. At this stage the images with tables and images without tables will be identified as a result of the table image classification task. For the table detection task the location of the table object in the image will be provided as a result. The detected table can then be extracted and sent to the next stage to address table structure detection. Several previous works have presented deep learning based solutions to accurately detect tables in images [85, 14, 12, 13] but there has been no previous work for table image classification.

We propose utilizing TableDet to address both table detection and table image classification in a single inference. As shown in Section 4.4.2 The TableDet model achieves the highest F1 scores for table detection on three public datasets (ICDAR 13 [68], ICDAR 17 [79] and ICDAR 19 [8]). Furthermore, for table image classification, it also accurately identifies all images with tables and significantly reduces the number of images without tables from being processed in the proposed pipeline.

At stage 2 of the pipeline we propose identifying the components of the detected table region as shown in Fig. 1.1. The table structure detection is a more complex task due to the multiple variations in the table layouts in documents published by different organizations.

The tables can appear with or without line separators between the cells [14]. Furthermore certain the information in the table can be presented in a hierarchically structured layout by utilizing merged cells (a common cell shared by multiple neighbouring cells) [16, 21]. In this chapter we propose a solution to address the table structure detection task in images of table regions (table images). This task can be decomposed into detecting rows and columns [56, 17] which can then be utilized to denote the locations of cells within the table. An alternative approach would be to directly detect the cells of the tables [14, 16, 21]. The methods presented by [56, 14, 16, 17] have not directly dealt with identifying hierarchical layouts of information. On the other hand, the solution presented by [21] has attempted to detect this hierarchical layout by detecting cells. To the best of our knowledge, there exists no prior work that detects rows and columns with the intention of providing more details about the hierarchical layout of information.

To address this task of table structure detection, we present TableStrDet, a deep learning based approach that detects and classifies rows and columns in table images. Utilizing the accurate locations provided by the TableDet model we can extract the tables from such images to develop the table images. In each table image we propose detecting and classifying columns as regular columns (columns without a merged cell) and irregular columns (group of regular columns with a shared merged cell). Similarly, we propose detecting and classifying rows as regular rows (rows without a merged cell) and irregular rows (group of regular rows with a shared merged cell). In addition to that we propose that the TableStrDet model utilizes one Cascade R-CNN [11] architecture to detect and classify the rows and another to detect and classify the columns in parallel. Thus, in a single inference TableStrDet can detect four classes of objects in a table image. Each Cascade R-CNN architecture is equipped with a deformable convolution backbone [55] and utilizes Complete IOU loss [18] to improve their object detection performance.

We describe the steps taken to develop the TableStrDet model, and the process of classifying rows and columns in Section 6.1. We describe the datasets utilized in our analysis in Section 6.2 and the training details in Section 6.3. We display our detection and classification analysis for rows and columns that are taken to develop the TableStrDet model in Section 6.4. We also provide details on the generalization potential of our proposed approach in Section 6.4. We provide our concluding statements in Section 6.5

6.1 System Model

In this section we describe the process of developing the TableStrDet model which detects rows and columns on table images. In addition, to detecting these table components

it also classifies the components into four classes; regular rows, irregular rows, regular columns, irregular columns. Multiple deep learning-based solutions have addressed the table detection task in images [85, 14, 17, 50]. However, we propose utilizing the detected table region provided by the TableDet model which we describe in Chapter 4 to develop such table images. For the task of table structure detection we determined that utilizing a table image which contains only the table region of a full page image and avoids the non-table sections of the image would help the TableStrDet model to focus on locating the rows and columns of the table as shown in [56, 16, 17].

6.1.1 Task Definition

The structure of the table can be decomposed into two classes or components : rows and columns as shown in [56, 17]. It can further be decomposed into a single class or component : cells as shown in [14, 16]. However, the tables within different datasheet (document) images have different layouts, based on the need to display information in a concise and non-redundant format. This difference is developed by the presence or absence of merged cells and the presence or absence of line separators between cells. We define a merged cell as multiple cells grouped into one cell that stretches across more than one row or column [21]. These merged cells indicate the common content between two or more rows or columns as shown in Fig 6.1. These merged cells are added in order to reduce the repetition of content among the rows or columns. Furthermore, it also lays out the information in a hierarchical format. In addition to the existence or absence of merged cells, the tables may or may not contain line separators between rows and columns to indicate these merged cells. Due to the large variety of layouts of different tables, we propose that it would be beneficial to detect rows and columns first in order to understand the layout of the content within the table.

Regular and irregular class definitions

We address the table structure detection task on table images as an object detection task where the objects to be detected are the rows and columns of the table. As shown in Fig. 6.2 (b) and Fig. 6.2 (c) we refer to the rows and columns that do not contain a merged cell (multiple cells grouped as one cell) as regular rows and regular columns, respectively. We refer to a group of regular columns that share a merged cell [21] which stretches across them as irregular column as shown in Fig. 6.2 (e). Similarly we refer to an irregular row as a group of regular rows that share a merged cell [21] that stretches across them as shown in Fig. 6.2 (d). Thus, along with detecting regular rows and regular

Site	11/7/2001 Flow		12/7/2001 Flow		13/7/2001 Flow		16/7/2001 Flow	
	Raw	Adj.	Raw	Adj.	Raw	Adj.	Raw	Adj.
A	1675	1639	1649	1606	1910	1889	1779	1732
B	84	81	69	68	98	98	65	62
C	827	806	856	835	918	890	979	961
D	509 ⋆	503 ⋆	946	934	1133	1116	1108	1096
E	1481	1439	1512	1468	1603	1553	1578	1538
F	557	545	548	527	643	624	603	582
G	1165	1136	1109	1084	1300	1270	1226	1210
H	948	929	862	848	843	829	1355	1307
I	898	872	881	870	837	825	886	871
J	494	476	506	488	477	451	460	443
K	675 ⋆	659 ⋆	723	708	680	665	727	716

(a)

label	max_s		
	initialization	1^{th} traversal	2^{nd} traversal
1	g_b	g_b	g_b
2	0	0	g_f
3	g_a	g_a	g_a
4	0	g_c	g_c
5	0	g_e	g_e
6	0	0	g_f
7	0	0	g_f
8	0	g_e	g_e
9	0	g_f	g_f

(b)

Source	Lead time (years)			
	1	2	3	4
	Absolute value of percentage difference between actual and projected values			
Projections of Education Statistics to 2017	†	0.7	1.1	1.4
Projections of Education Statistics to 2018	0.4	0.7	0.8	1.1
Projections of Education Statistics to 2019	#	0.1	0.2	†
Projections of Education Statistics to 2020	0.2	0.4	†	†
	Mean absolute percentage error			
Example	0.2	0.5	0.7	1.3

(c)

File	Size (bytes)	Execution times (in seconds)		
		ZLIB/deflate	bzip2	clazz
AudioClip.class	233	0·049 0·019	0·577 0·385	0·049 0·025
Component.class	24,622	0·368 0·088	1·340 0·577	0·338 0·370
Enumeration.class	261	0·052 0·024	0·538 0·373	0·052 0·025
HashTableEntry.class	630	0·053 0·026	0·563 0·376	0·058 0·030
Integer.class	3,733	0·105 0·032	0·747 0·417	0·103 0·070
Object.class	1,452	0·057 0·026	0·598 0·381	0·067 0·037

(d)

Figure 6.1: **Merged cells (highlighted with blue boxes) in multiple table types**

columns we propose detecting irregular rows and irregular columns in order to provide more details on the layout of the information in the table.

Psychosomatic Symptoms	How often have you taken medicine or tablets?		
	No	Occasionally	Frequently
Headache	228	163	99
Stomach-ache	421	49	16
Sleeping Difficulties	458	16	7
Nervousness	449	20	14

(a)

Psychosomatic Symptoms	How often have you taken medicine or tablets?		
	No	Occasionally	Frequently
Headache	228	163	99
Stomach-ache	421	49	16
Sleeping Difficulties	458	16	7
Nervousness	449	20	14

(b)

Psychosomatic Symptoms	How often have you taken medicine or tablets?		
	No	Occasionally	Frequently
Headache	228	163	99
Stomach-ache	421	49	16
Sleeping Difficulties	458	16	7
Nervousness	449	20	14

(c)

Psychosomatic Symptoms	How often have you taken medicine or tablets?		
	No	Occasionally	Frequently
Headache	228	163	99
Stomach-ache	421	49	16
Sleeping Difficulties	458	16	7
Nervousness	449	20	14

(d)

Psychosomatic Symptoms	How often have you taken medicine or tablets?		
	No	Occasionally	Frequently
Headache	228	163	99
Stomach-ache	421	49	16
Sleeping Difficulties	458	16	7
Nervousness	449	20	14

(e)

Figure 6.2: **Original table image (a). Table image with regular rows (b) and regular columns (c) highlighted with blue boxes. Table image with irregular rows (d) and regular columns (e) highlighted with red boxes.**

Detecting groups of components vs detecting cells

Instead of detecting only the merged cell in this study, we propose detecting the entire group of rows and columns attached to the merged cell for two reasons. Firstly, by capturing the group of regular rows and columns that are part of irregular rows and columns respectively, it becomes easier to provide and denote the hierarchical details of which rows and columns share a common merged cell. Secondly, we determined that if we only detect the merged cells along with the regular rows and columns, then the object detection architecture would try to detect cells along with rows and columns which would be redundant. Thus, we propose not only to detect rows and columns but to also detect and classify them

as regular rows, irregular rows, regular columns and irregular columns to address table structure detection in table images.

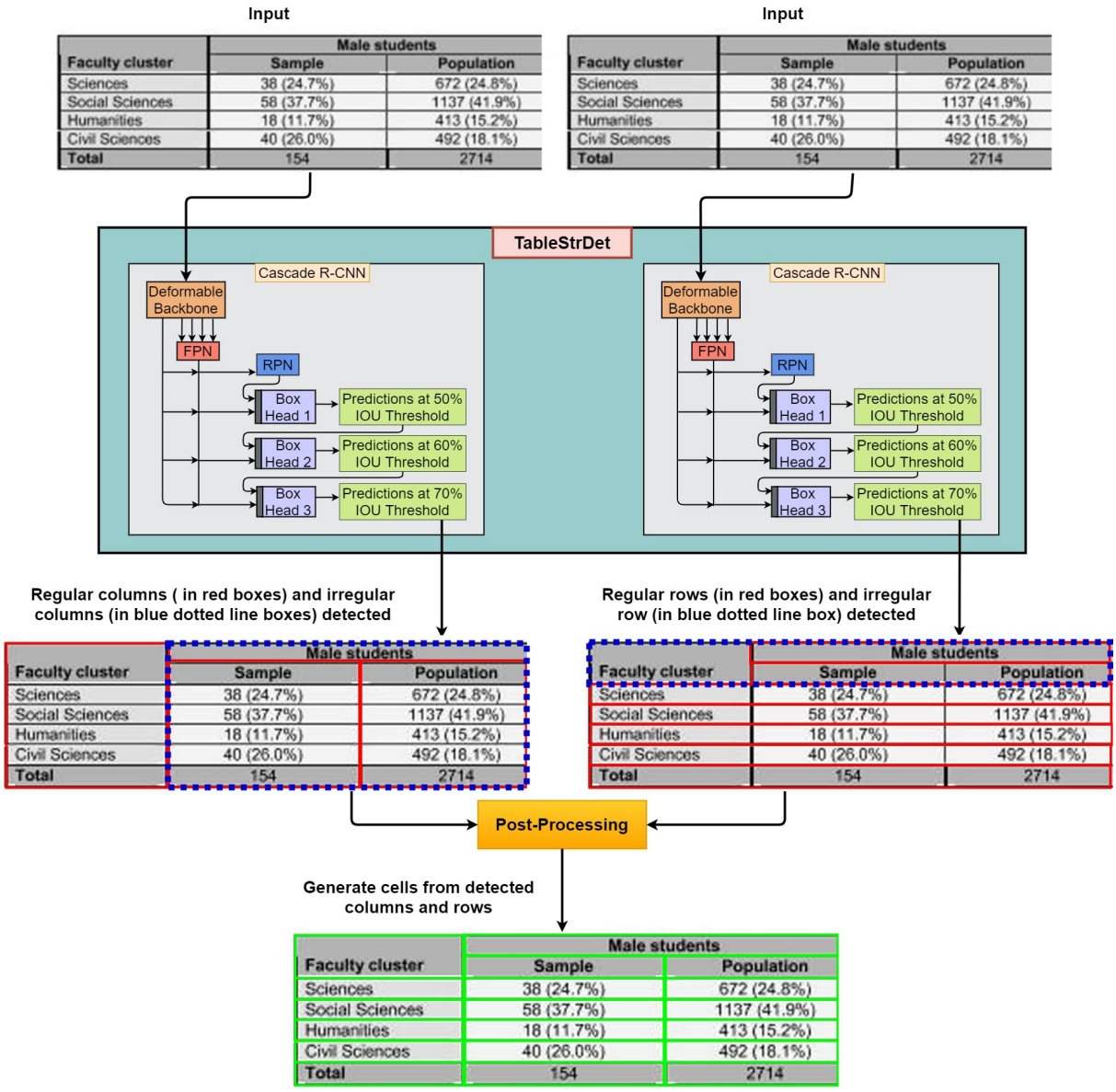


Figure 6.3: **TableStrDet model comprises of two Cascade R-CNN architectures each with deformable backbones and CIOU loss to address table structure detection in table images.**

6.1.2 TableStrDet architecture

Deep learning-based techniques achieve state-of-the-art performances on tasks such as image classification and object detection due to the generalization ability of the same architecture to detect different types of objects such as natural objects [11] or objects in datasheet (document) images [14]. In this work, we utilize a deep learning based object detection approach to recognize the structure of a table in a table image. This image will contain only the detected table region of a full page datasheet (document) image. Furthermore, we determined that the structure of the table can be decomposed into four classes of objects. Instead of detecting only rows and columns, we propose detecting irregular and regular rows along with irregular and regular columns in order to capture the merged cells within the structure. By identifying these irregular rows and irregular columns, the object detection architecture would be able to provide more details about the hierarchical layout of information in the table.

Thus, we put forward TableStrDet, a deep learning based approach that utilizes an object detection architecture to accurately identify objects of these four classes in table images. We employ a Cascade-RCNN [11] object detection architecture to construct the TableStrDet model. The Cascade R-CNN architecture comprises of four components as shown in Fig. 6.3: backbone, Feature Pyramid Network (FPN), Region Proposal Network (RPN) and three Region of Interest or box heads. In this work, as a backbone to act as the feature extractor we utilize the ResNet [44] architecture. This architecture comprises of four convolution blocks as shown in Fig. 3.1. Each block contains a series of convolution layers stacked on top of each other to provide feature maps at different scales. These convolution layers have a fixed receptive field that operates as a sliding window over each section of the image. They provide rich features to the top layers of the Cascade R-CNN architecture that reside in the RPN and box heads as shown in Fig. 3.2. However, fixed receptive fields cannot capture all the features for objects that appear with different scales and orientations.

As rows and columns appear with different scales we employ deformable convolution [55] which have an adaptive receptive field to capture variances in features of such objects. We refer to the receptive field that operates as a sliding window over each section of the image as a sampling matrix. The deformable convolution operation [55] continues to utilize all parameters of the standard convolution operation. These parameters include the image / input feature map (x), the central location of the sampling matrix (p_0) and the neighbouring locations on the matrix (p_i). It also continues to update the weights (w) of this matrix to generate the output feature map (y). Furthermore, it captures offsets (Δp_i) of the object features as shown in (6.1). These offsets store the weights at each location

of the sampling matrix when the rows and columns have multiple scales. Thus, we utilize a deformable convolution ResNet architecture in this work to extract multi-scale features from table images.

$$y(p_0) = \sum_{p_i \epsilon C} w(p_i).x(p_0 + pi + \Delta p_i) \tag{6.1}$$

.

$$IoU(B, B^{gt}) = \frac{B \cap B^{gt}}{B \cup B^{gt}} \tag{6.2}$$

The Cascade R-CNN architecture utilizes three box heads to refine the bounding boxes for each image [11] at multiple Intersection Over Union (IOU) values [114]. The IOU value is measured as the intersection over union of proposed bounding box ($B = (x_1, y_1, x_2, y_2)$) generated by the architecture and ground truth ($B^{gt} = (x_1^{gt}, y_1^{gt}, x_2^{gt}, y_2^{gt})$) bounding box as shown in (6.2).

Furthermore, each box head receives multi-scale rich feature maps from the FPN and the final layer of the backbone along with proposal bounding boxes for refinement as shown in Fig. 3.2 and Fig. 6.3. These feature maps will aide the box heads and the RPN to detect objects at multiple scales. In addition to that, the first box head refines the bounding boxes it receives from the RPN at an IOU value greater than 50%. These improved boxes are provided as proposals to the second box head where the bounding boxes greater than 60% IOU value are refined and are then sent to the third box head where the boxes above 70% IOU value are refined and provided as the final output.

At each box head, the bounding boxes are refined by calculating the bounding box regression loss to measure the difference between the ground truth and proposed bounding boxes. In addition to that, a cross entropy loss is also calculated at each box head to measure the difference between the predicted and ground truth object class labels. Both losses are calculated together as a multi-task loss as formulated in (6.3).

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{6.3}$$

IOU-based regression loss

The standard bounding box regression loss utilized by each box head of the Cascade R-CNN architecture [11] is the $l1$ loss. However, the bounding box regression performance of

each box head is evaluated in terms of an IOU value. Furthermore, rows and columns can appear with different scales in table images. Utilizing geometric factors such as distance between boxes and aspect ratio as parameters in the regression loss calculation can be beneficial towards refining the bounding boxes for objects with multiple scales shown as shown by Zheng et al. [18]. Furthermore, to bridge this gap between the metrics for loss calculation and evaluation, we propose utilizing Complete IOU (CIOU) loss [18] at each box head to refine the bounding box predictions from the TableStrDet model. CIOU loss [18] operates with the same parameters as Distance IOU (DIOU) loss, but also considers the aspect ratio of the objects to improve on DIOU loss calculation.

$$v = \frac{4}{\pi^2}(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h})^2 \tag{6.4}$$

$$\alpha = \frac{v}{(1 - IoU + v')} \tag{6.5}$$

$$L_{CIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \tag{6.6}$$

Along with utilizing the IOU value as a penalty term, DIOU loss considers the distance between the center points of the two boxes ($\rho(.)$) and the diagonal length of the largest polygon box that can cover both boxes (c). CIOU loss adds two parameters to include the aspect ratio of the boxes in the calculation; the consistency of the aspect ratio (v) and positive trade-off parameter (α) as given in (6.4) and (6.5) respectively. These parameters are added to the existing parameters of DIOU loss to calculate CIOU loss as shown in (6.6). To accurately detect and classify the rows and columns we employ CIOU loss at all three box heads of the Cascade R-CNN architecture which is utilized to develop the TableStrDet model.

Rows generally appear with a horizontal alignment while columns appear with a vertical alignment [15] within the table. Thus, our proposed TableStrDet model consists of two Cascade R-CNN architectures each detecting and classifying either rows or columns in the same table image as shown in Fig. 6.3. Each architecture utilizes a deformable backbone and CIOU loss at each box head to improve bounding box predictions. As shown in Fig 6.3, we employ one Cascade R-CNN architecture to detect the rows (regular and irregular)

and the second Cascade R-CNN architecture to detect the columns (regular and irregular) in the table images. Both architectures work together in parallel to provide these results. The final result of the TableStrDet model are the locations of regular and irregular rows along with regular and irregular columns with the motivation of gaining more insights about the table structure. Hence, the presented TableStrDet model detects four classes of objects in a single inference.

We display the inference process of TableStrDet in Fig. 6.3. Since there are two Cascade R-CNN architectures working in parallel within the TableStrDet model, we provide two copies of input table images to the TableStrDet model. The input table image in Fig. 6.3 has objects of all four classes. Thus, the TableStrDet model provides the locations of each of the four object classes. We denote the detected irregular rows and columns with blue dotted boxes while the detected regular rows and columns are denoted with red boxes to indicate the output of the TableStrDet model. Furthermore, as a post-processing step we utilize the intersecting lines of the detected rows and columns to generate cell locations as indicated in Fig. 6.3.

The cell locations display not only the locations of regular cells (cells not stretching over multiple cells) but also the locations of merged cells. The irregular rows and irregular columns consider the merged cell and the regular rows and regular columns, respectively. Thus, the intersecting lines developed by these four detected object classes presents an easier pathway to locate the merged cells in the table as shown in the final result in Fig. 6.3. We determined that by detecting these four object classes the TableStrDet model can accurately identify the hierarchical layout of data within the table image irrespective of the presence or absence of line separators between the rows and columns.

6.2 Datasets

In this section we describe the datasets utilized to develop the TableStrDet model and compare its table structure detection performance with state-of-the-art methods. We utilize two datasheet (document) image datasets in this work; ICDAR 2013 TS [68] and TabStructDB TS [56]. Both datasets consist of table images which are the table regions of each full page image. Each image is also accompanied with a set of bounding boxes (annotations) to denote the location of the rows and columns of the table. We manually corrected the row and column annotations of each image to ensure that the rows and columns within the image are aligned with each other and they do not capture merged cells. We refer to these rows and columns as regular rows and regular columns, respectively. Moreover, we develop annotations for irregular rows and irregular columns as well.

An irregular row is annotated by drawing a single bounding box over a group of regular rows and the merged cell that stretches over these regular rows. Thus an irregular row annotation captures the regular rows as well as the merged cell shared by them. Similarly we develop annotations for irregular columns which group together multiple regular columns and include the merged cell that stretches over these regular columns.

Table 6.1: **Datasets utilized to train and test TableStrDet**

Dataset	Train	Test
ICDAR 2013 TS	125	31
TabStructDB TS	731	350
Total	856	381

The ICDAR 2013 TS dataset [68] includes 156 table images. We maintain the same train and test splits as Siddiqui et al. [56] to train and compare the table detection performance of our presented approach with the state-of-the-art solution. Thus, we train the TableStrDet on 125 and evaluate its performance on 31 table images from this dataset. We also utilize the TabStructDB TS [56] which includes 731 training images and 350 test images of tables from the ICDAR 2017 POD dataset [79]. We maintain the same train and test split as Siddiqui et al. [56] for this dataset as well. For images of both datasets we have ensured to annotate all four object types; regular rows, irregular rows, regular columns and irregular columns.

6.3 Training Details

In this work, we propose detecting and classifying rows as well as columns to address the task of table structure detection in table images. To the best of our knowledge, the state-of-the-art method for row and column detection on these two available datasets from Table 6.1 is the DeepTabStr [56] model. The DeepTabStr model only detects rows and columns and does not classify them into regular and irregular classes. Thus, we train the DeepTabStr model on the two datasets from Table 6.1 to evaluate and fairly compare the detection performance our presented approach with the state-of-the-art solution.

The TableStrDet model utilizes the Cascade R-CNN [11] architecture with deformable ResNet-50 [44] backbone. The DeepTabStr model utilizes the deformable FPN [86] architecture with a deformable ResNet-101 [44] backbone. We demonstrate the impact of

training these models with two architectures against single architectures. We refer to the model with two architectures as separate version and the model with one architecture as combined version. We analyze the impact of utilizing the separate and combined versions of these two solutions on their table structure detection performances.

The separate versions comprise of one architecture trained to detect regular and irregular rows while the other architecture is trained to detect regular and irregular columns. The combined versions comprise of one architecture that is trained to detect all four object classes. The separate versions are individually trained for 60 epochs whereas the combined versions are trained for 70 epochs. Both versions of each model are trained on the training set of the ICDAR 2013 TS and TableStructDB datasets as shown in Table 6.1.

The final inference of both models irrespective of their versions are the locations and classes of the four object types. TableStrDet has been developed with the Detectron2 library [1]. We train the separate and combined versions of TableStrDet with a batch size of 2 images. In addition to that, Group Normalization [140] enhances the detection performance of object detection architectures such as Cascade R-CNN when the training batch size is small. Thus, we utilize Group Normalization in the architecture of both versions of TableStrDet instead of the standard Batch Normalization [139] to improve their detection performance. We utilize a learning rate of 0.002 with a SGD optimizer to train the models. All versions of TableStrDet and DeepTabStr have been trained on a single NVIDIA Tesla V100 GPU with 32GB RAM.

6.4 Results

In this section we present the improvements in table structure detection on table images that are brought on by our proposed TableStrDet model. The model approaches the task of table structure detection as an object detection task. The model detects four classes of objects; regular rows, irregular rows, regular columns and irregular columns. It detects all four object classes in a single inference to provide the locations of content in a table image as shown in Fig. 6.3. To the best of our knowledge, the state-of-the-art solution for row and column detection on these two datasets is the DeepTabStr [56] model. The DeepTabStr model only detects rows and columns and does not classify them into regular and irregular classes. Thus, we train the DeepTabStr model on the two available datasets and to evaluate and fairly compare the detection performance our presented solution with the state-of-the-art solution.

[1] https://github.com/facebookresearch/detectron2

We utilize the ICDAR 2019 Table Detection and Recognition competition [8] evaluation metric to evaluate the table structure detection performances that are achieved by TableStrDet and the state-of-the-art DeepTabStr method on the two available test sets. As per this metric the F1 score at four IOU values $(60, 70, 80, 90)$ are calculated. Following upon this, the corresponding IOU values are used as the weights to each F1 score. The product of the weights and the F1 score are then added to develop the weighted average F1-score (W. Avg F1).

We analyze multiple deep learning based architectures such as Mask R-CNN [52] and Cascade R-CNN [11] to develop the TableStrDet model. In addition, we evaluate the impact of utilizing a deformable convolution [55] backbone instead of standard convolution backbone. Furthermore, we examine the detection performance improvement brought on by using separate versions of the models over combined versions as well. The separate version uses two architectures where one architecture detects and classifies columns and another detects and classifies rows. In the combined version, a single architecture detects all four objects. For instance the separate version of TableDetStr utilizes two Cascade R-CNN architectures each with deformable ResNet-50 backbones and CIOU loss where one architecture detects and classifies rows while the other detects and classifies columns in parallel. For the separate version of DeepTabStr, two deformable FPN architectures each with deformable ResNet-101 backbones are utilized. In the combined versions of TableStrDet and DeepTabStr, we utilize a single Cascade R-CNN and a single FPN architecture respectively to detect and classify rows and columns.

This analysis is conducted on the ICDAR 2013 TS and TabStructDB TS datasets as shown in Table 6.2 and Table 6.3 respectively. All models are trained to detect regular rows, regular columns, irregular rows and irregular columns. While the models detect four classes of objects we measure their detection performance on how well they detect and classify rows and columns as shown in Table 6.2 and Table 6.3. We infer that the TableStrDet model should utilize two Cascade R-CNN architectures each with deformable backbones and CIOU loss to accurately detect and classify rows and columns separately. In our analysis, we determine that the separate version of TableStrDet is able to outperform the separate version DeepTabStr in detecting and classifying both rows and columns. Furthermore, we measure how efficient the separate version of TableStrDet is over DeepTabStr on detecting objects of each of the four classes in Table 6.4. We also measure the generalization capability of both solutions as shown in Table 6.5. The results of our approach can be seen in Fig. A.11, Fig. A.12 and Fig. A.13.

Table 6.2: **Table Structure Detection results comparison with state-of-the-art models on ICDAR 13 test set**

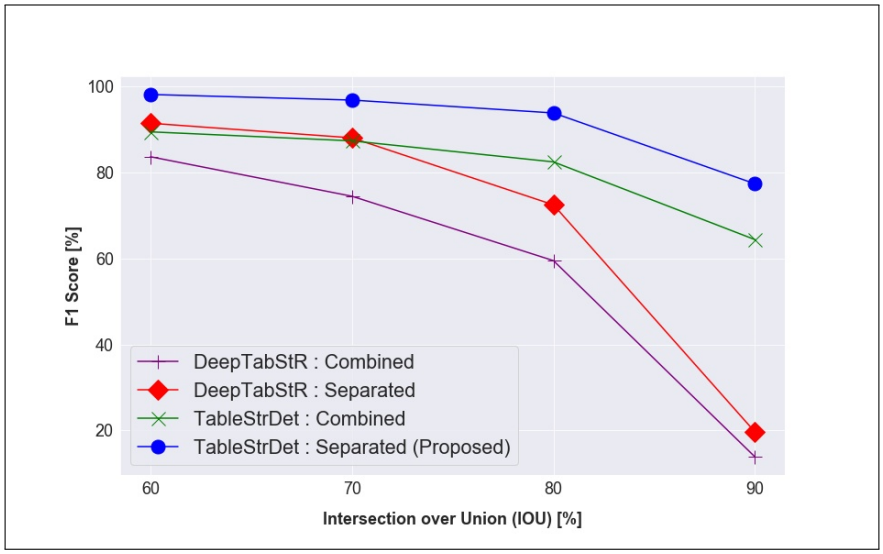
Model	Version	Row (Regular and Irregular)					Column (Regular and Irregular)				
		F1 score at each IoU				W.Avg	F1 score at each IoU				W.Avg
		60%	70%	80%	90%	F1 %	60%	70%	80%	90%	F1 %
DeepTabStR [56]	Combined	83.7	74.5	59.5	13.8	54.1	91.7	87.4	80.8	38.9	71.9
	Separated	91.5	88.1	72.5	19.6	64.1	91.3	89.7	87.1	67.1	82.6
Mask-RCNN	Combined	85.9	76.6	64.0	14.6	56.5	90.7	88.0	73.9	45.8	72.1
	Separated	90.6	83.4	73.7	23.8	64.4	90.8	90.0	86.5	72.0	84.0
Mask-RCNN Deform	Combined	83.6	81.2	74.2	23.5	62.5	89.2	87.8	85.1	45.9	74.8
	Separated	95.8	92.8	80.0	40.9	74.4	91.8	90.8	89.0	75.6	86.0
Cascade-RCNN	Combined	87.8	86.5	81.7	54.3	75.8	90.5	89.3	86.8	80.0	86.1
	Separated	98.0	94.8	91.8	75.6	88.9	92.0	90.7	88.8	81.0	87.6
Cascade-RCNN Deform	Combined	89.7	88.6	83.5	57.3	78.1	91.1	90.5	88.5	80.7	87.2
	Separated	**98.6**	95.6	93.7	75.3	89.6	92.0	91.2	90.0	82.8	88.5
Ours(TableStrDet)	Combined	89.5	87.4	82.5	64.4	79.6	91.1	90.5	89.2	82.3	87.8
	Separated	98.2	**96.9**	**93.9**	**77.4**	**90.5**	**92.1**	**91.7**	**89.6**	**85.2**	**89.6**

ICDAR 13 Dataset

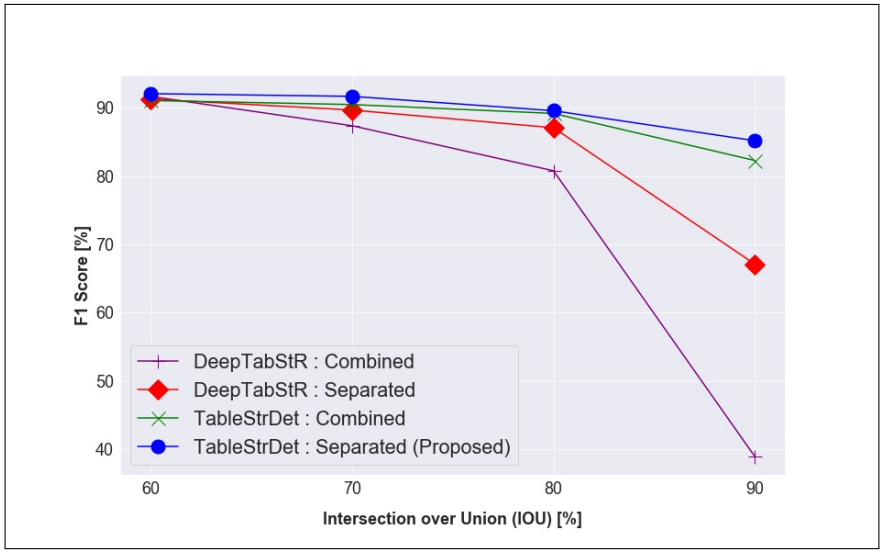
As shown in Table 6.2, we evaluate the performance of DeepTabStr [56] and TableStrDet on the ICDAR 2013 TS test set [68]. Both models are trained on the training set of the ICDAR 2013 TS to detect four classes of objects. We evaluate the impact of utilizing the separate and combined versions of both TableDetStr and DeepTabStr. The separate versions outperform the combined versions especially on the task of detecting and classifying rows. The separate version of TableStrDet outperforms the separate version of DeepTabStr with a 26.4% and 7% increase in performance in terms of weighted average F1 score for rows and columns, respectively. At each IOU value the separate version of TableStrDet achieves a higher F1 score than the separate version of DeepTabStr in detecting and classifying rows and columns.

In Fig. 6.4 we display the detection results that are achieved by the separate and combined versions of TableStrDet and DeepTabStr. As shown in Fig. 6.4 (a), the separate version of TableStrDet achieves a consistently higher F1 score in detecting and classifying rows that DeepTabStr. In Fig. 6.4 (b), we show that TableStrDet can achieve a higher F1 score at all four IOU values especially at 90% IOU value for detecting and classifying columns. At 90% IOU value the separate version of TableStrDet achieves a 18.1% and 57.8% increase in detecting and classifying rows and columns over the separate version of DeepTabStr.

The separate version of the TableStrDet achieves the highest F1 score as compared to the other models at all IOU values for both rows and columns except at 60% IOU value



(a) Results for detecting regular and irregular rows



(b) Results for detecting regular and irregular columns

Figure 6.4: **Comparing detection performances of separate versions of TableStrDet and DeepTabStR on ICDAR 2013 TS test set**

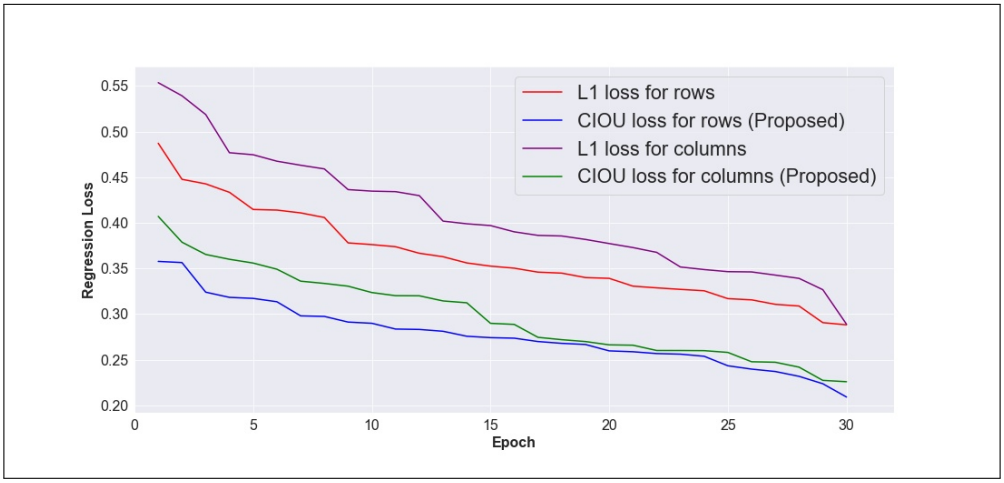


Figure 6.5: **L1 loss and CIOU loss comparison at each training epoch achieved by separate versions of TableStrDet for detecting and classifying rows and columns on ICDAR 2013 TS test set**

for rows. Furthermore, the separate versions of all models displayed in Table 6.2 achieve higher F1 scores and weighted average F1 scores for detecting and classifying rows and columns. Thus, we infer that the separate versions of the models offer a more efficient solution for this problem.

In addition we also display the training improvements provided by utilizing CIOU loss as a bounding box regression loss in Fig. 6.5. We noted the average regression loss that is achieved by the separate version of TableStrDet when $l1$ loss or CIOU loss is used in the architecture. It is worth noting that the architectures which are trained to detect and classify rows and columns with CIOU loss converged faster than the architectures which are trained with $l1$ loss. As shown in Fig. 6.5 and Table 6.2 utilizing CIOU loss for bounding box regression in TableStrDet improves its row and column detection performance.

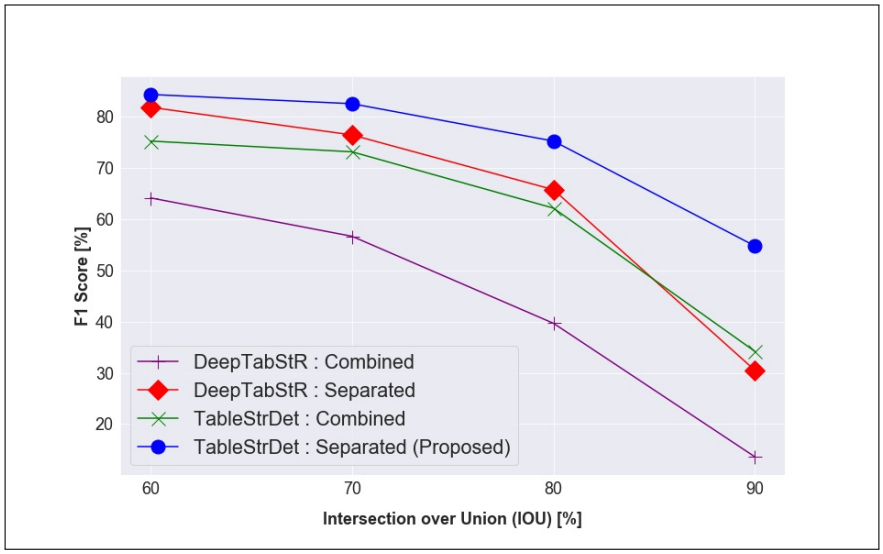
We further evaluate how well the separate versions of the TableStrDet and DeepTabStr can detect each of the four object classes in Table 6.4. TableStrDet is able to achieve the highest weighted average F1 score for all four object classes as shown in Table 6.4. It achieves a 99.8% and 85.2% weighted average F1 score for irregular rows and columns, respectively. At 90% IOU value TableStrDet achieves greater than 9.9% improvement in detecting and classifying both rows and columns over DeepTabStr. Thus, it can be concluded that TableStrDet can improve the detection performance of not only regular rows and regular columns but also irregular rows and irregular columns.

Table 6.3: **Table Structure Detection results comparison with state-of-the-art models on TabStrDB test set**

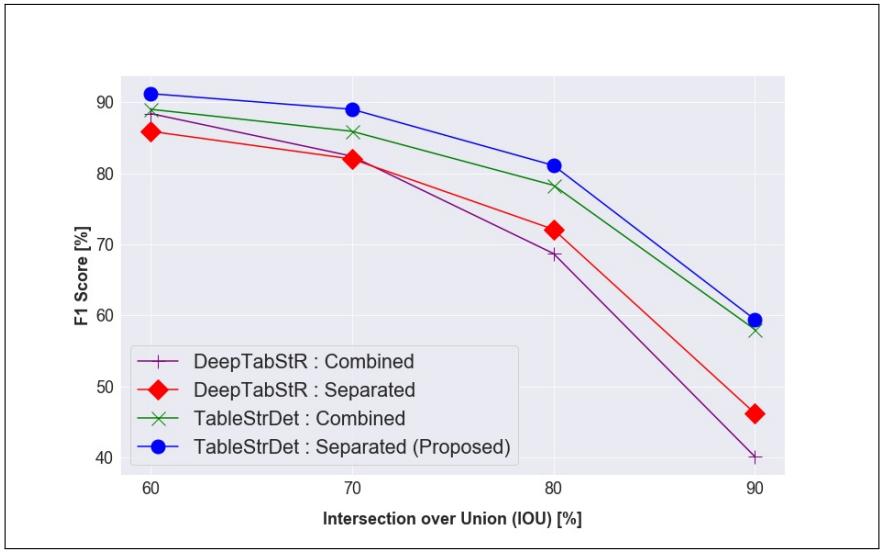
Model	Version	Row (Regular and Irregular)					Column (Regular and Irregular)				
		F1 score at each IoU				W.Avg	F1 score at each IoU				W.Avg
		60%	70%	80%	90%	F1 %	60%	70%	80%	90	F1 %
DeepTabStR [56]	Combined	64.2	56.7	39.7	13.6	40.8	88.4	82.4	68.7	40.1	67.2
	Separated	81.9	76.5	65.8	30.5	60.9	85.9	82.0	72.1	46.2	69.4
Mask-RCNN	Combined	62.0	56.4	43.3	15.6	41.8	87.4	82.0	71.3	40.9	67.9
	Separated	78.3	75.2	64.1	35.3	60.9	87.7	84.6	76.5	51.9	70.7
Mask-RCNN Deform	Combined	62.4	55.9	44.4	16.8	42.4	87.0	83.1	70.8	40.1	67.7
	Separated	80.1	77.2	68.2	38.2	61.4	89.7	86.4	80.4	58.2	74.3
Cascade-RCNN	Combined	69.5	67.3	58.7	30.7	54.5	88.6	86.0	79.0	55.0	75.3
	Separated	82.0	78.6	71.0	47.0	67.8	90.1	87.9	80.7	58.2	77.5
Cascade-RCNN Deform	Combined	71.6	69.1	61.2	33.9	56.9	88.0	85.4	77.8	57.0	75.4
	Separated	82.1	79.8	72.7	51.3	69.8	91.2	88.4	79.9	59.1	77.9
Ours(TableStrDet)	Combined	75.3	73.2	62.2	34.2	59.0	89.0	85.9	78.3	58.0	76.1
	Separated	**84.4**	**82.6**	**75.3**	**54.8**	**72.7**	**91.2**	**89.0**	**81.1**	**59.4**	**78.5**

Table 6.4: **Table Structure Detection results comparison for irregular and regular classes of rows and columns on ICDAR 13 and TabStrDB test sets**

Dataset	Model	Row						Column					
		Class	F1 score at each IoU				W.Avg	Class	F1 score at each IoU				W.Avg
			60%	70%	80%	90%	F1 %		60%	70%	80%	90	F1 %
ICDAR 13 (Train and Test Set)	DeepTabStr	Regular	**97.8**	92.7	79.0	33.0	72.1	Regular	**99.2**	96.1	92.0	74.2	89.0
	Ours(TableStrDet)		96.9	**94.5**	**88.4**	**54.6**	**81.4**		**99.2**	**98.4**	**94.3**	**85.4**	**93.6**
	DeepTabStr	Irregular	88.0	88.0	88.0	61.4	80.0	Irregular	85.0	85.0	85.0	75.3	82.1
	Ours(TableStrDet)		**99.8**	**99.8**	**99.8**	**99.8**	**99.8**		**85.2**	**85.2**	**85.2**	**85.2**	**85.2**
TabStructDB TS (Train and Test Set)	DeepTabStr	Regular	92.2	87.3	73.1	30.1	67.4	Regular	95.8	91.9	78.6	49.1	76.3
	Ours(TableStrDet)		**93.5**	**90.6**	**78.3**	**41.2**	**73.1**		**97.6**	**94.0**	**82.3**	**54.5**	**79.7**
	DeepTabStr	Irregular	75.4	71.1	68.9	45.3	63.7	Irregular	80.6	76.5	71.9	51.9	68.7
	Ours(TableStrDet)		**82.0**	**81.1**	**79.1**	**74.3**	**78.7**		**87.4**	**86.7**	**82.5**	**66.3**	**79.6**



(a) Results for detecting regular and irregular rows



(b) Results for detecting regular and irregular columns

Figure 6.6: **Detection performances of separate versions of TableStrDet and DeepTabStR on TabStructDB TS test set**

TabStuctDB Dataset

We perform a similar analysis on the TabStructDB TS dataset to evaluate the performance improvement achieved by our proposed approach. All models are trained on the TabStructDB TS training set for this analysis. The separate version of TableStrDet achieves the highest weighted average F1 score for both detecting and classifying both rows and columns as shown in Table 6.3. It outperforms the separate version of DeepTabStr by 11.8% and 9.1% on rows and columns respectively in terms of weighted average F1 score. It also achieves the highest F1 score on all four IOU values for both detecting and classifying rows and columns as shown in Table 6.3.

We display the results that the separate and combined versions of TableStrDet and DeepTabStr achieve in Fig. 6.6. As shown in Fig. 6.6 the separate versions of both solutions outperform the combined versions. Furthermore, at each of the four IOU values, the separate version of TableStrDet achieves a higher F1 score in detecting and classifying rows and columns as shown in Fig. 6.6 (a) and Fig. 6.6 (b), respectively.

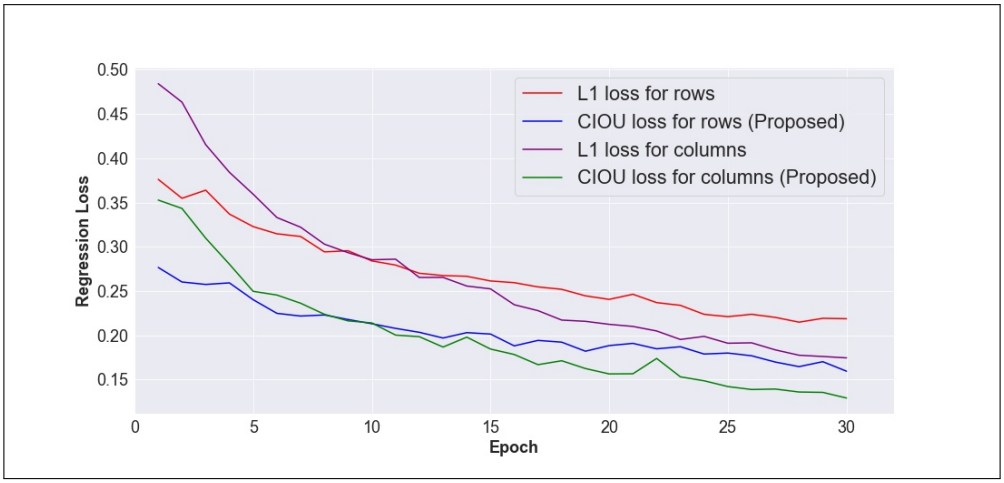


Figure 6.7: **Comparison of L1 loss and CIOU loss at each training epoch achieved by separate versions of TableStrDet for detecting and classifying rows and columns on TableStructDB test set**

In addition, we also display the quicker convergence enhancement provided by utilizing CIOU loss in the TableStrDet architecture in Fig. 6.7. Employing CIOU loss for bounding box regression not only helps TableStrDet converge faster but also improves the detection capability of the model to detect and classify rows and columns as indicated in Table 6.3.

Similar to our analysis on ICDAR 13 test set, we conclude that the separate versions of the models are more efficient solutions for this task on the TabStructDB TS test set as shown in Table 6.3 and Fig. 6.6. The separate version of TableStrDet also achieves the highest weighted average F1 score for not only regular and irregular rows but also regular and irregular columns as shown in Table 6.4. TableStrDet also achieves greater than 3% weighted average F1 score in detecting objects of all four classes over DeepTabStr. As shown Table 6.4 in at 90% IOU value it achieves greater than 5.4% improvement in detecting objects of all four classes. Furthermore, while the combined version of TableStrDet provides results in 2 seconds, the separate version takes just an additional 0.2 second to provide more accurate results.

Cross dataset

As shown in Tables 6.2, 6.3, and Figs. 6.4, Fig. 6.6 utilizing separate version of TableStrDet provides high quality rows and columns detection results. To measure the generalization capability of our proposed approach, we conduct the experiment presented by Siddiqui et al [56] as shown in Table 6.5. We utilize the separate version of TableStrDet and the separate version of DeepTabStr to measure the generalization potential of both solutions. As we have two datasets in this work, the models are trained on one dataset and tested on the other dataset to evaluate how well the models generalize to the test dataset. For instance, we train the models on ICDAR 2013 TS training set and test their rows and columns detection performance on the TabStructDB TS test set and TabStructDB TS complete set. Similarly as shown in Table 6.5, we train the models on TabStructDB TS training set and test their detection performance on the ICDAR 2013 TS test set and ICDAR 2013 TS complete set. The models are trained to detect and classify each object in the image into one of the four object classes; regular rows, irregular rows, regular columns and irregular columns.

Furthermore, the table images in both datasets appear in different types of documents. The tables in images from ICDAR 2013 TS appear in government documents while the tables in images from TabStructDB TS appear in scientific journals. Thus, we conduct this test to evaluate which method would perform better when it is exposed to unseen table images such as tables from datasheets. It is worth noting that the ICDAR 2013 TS training set contains only 125 table images while the TabStrucDB test set and TabStructDB TS complete set contain 350 and 1081 table images respectively. Due to the lower number of training images and the difference in structures of tables between ICDAR 2013 TS and TabStructDB TS dataset the models provide lower results especially in the case of detecting rows. However, in the case of detecting columns the models achieve greater

Table 6.5: **Table Structure Detection results comparison for irregular and regular classes of rows and columns on ICDAR 13 and TabStrDB test sets**

model	Training Set	Test Set	Row					Column				
			F1 score at each IoU				W.Avg	F1 score at each IoU				W.Avg
			60%	70%	80%	90%	F1 %	60%	70%	80%	90%	F1 %
DeepTabStr	ICDAR 2013 TS training set	TabStructDB TS test set	49.3	41.4	29.8	11.3	30.9	76.3	68.9	54.0	27.2	53.9
Ours(TableStrDet)			**57.3**	**47.5**	**34.7**	**20.9**	**38.1**	**80.5**	**75.3**	**61.7**	**39.7**	**62.0**
DeepTabStr		TabStructDB TS complete set	55.6	46.3	30.2	10.7	33.2	78.3	68.5	56	33.9	56.7
Ours(TableStrDet)			**63.1**	**55.2**	**39.3**	**20.2**	**42.1**	**83.6**	**77.1**	**66.4**	**48.0**	**66.8**
DeepTabStr	TabStructDB TS training set	ICDAR 2013 TS test set	82.9	72.3	60.3	12.3	53.2	81.5	76.7	72.2	56.8	70.5
Ours(TableStrDet)			**90.8**	**81.0**	**69.6**	**40.6**	**67.8**	**85.8**	**82.7**	**77.7**	**69.3**	**78.0**
DeepTabStr		ICDAR 2013 TS complete set	85.0	78.6	60.4	23.5	58.5	**86.0**	78.0	62.1	43.7	65.1
Ours(TableStrDet)			**89.1**	**84.5**	**73.9**	**40.9**	**69.5**	**86.0**	**80.4**	**72.3**	**53.8**	**71.3**

than 50% weighted average F1 scores. TableStrDet outperforms DeepTabStr at all IOU values for both the test set and complete set of TabStructDB TS despite the lower number of training images. In the second case, the models are trained on the TabStructDB TS training set which contains 731 table images. In this case the two test sets are the test set of ICDAR 2013 TS and complete set of ICDAR 2013 TS which are comprised of only 31 and 156 table images. Thus, TableStrDet is able to achieve greater than 67% weighted average F1 scores for detecting and classifying both rows and columns.

At all IOU values, TableStrDet achieve higher F1 scores as compared to DeepTabStr for all four test sets except at 60% IOU value for ICDAR 2013 TS complete set where the two methods share the same F1 score. When trained on the ICDAR 2013 TS training set, it achieves greater than 7.2% improvement in weighted average F1 score over DeepTabStr on both the test set as well as the complete set of the TabStructDB TS dataset. Similarly, when TableStrDet is trained on TabStructDB TS training set it achieves achieves greater than 6.2% improvement in weighted average F1 score over DeepTabStr for both the test set as well as the complete set of the ICDAR 2013 TS dataset.

Thus, we determine that the separate version of TableStrDet can generalize better to unseen table images as compared to DeepTabStr. Thus, we infer that TableStrDet model with two Cascade R-CNN architectures each with a deformable backbone and CIOU loss can not only improve row and column detection on two public test sets but can also have a higher generalization potential than the state-of-the-art solution.

6.5 Conclusion

In this chapter, we propose TableStrDet, a deep learning-based approach to detect and classify rows and columns in table images. TableStrDet comprises of one Cascade

R-CNN architecture to detect and classify rows as regular rows and irregular rows and another Cascade R-CNN architecture to detect and classify columns as regular columns and irregular columns to solve the table structure detection task. Each Cascade R-CNN architecture employs a deformable convolution backbone and utilizes Complete IOU loss to improve its detection performance. These enhancements enable TableStrDet to achieve 90.5% and 89.6% weighted average F1 score under the ICDAR 2013 TS test set for rows and columns, respectively, surpassing the state-of-the-art solution on this test set. On the TabStructDB TS test set, TableStrDet achieves the highest weighted average F1 score for rows and columns with 72.7% and 78.5%, respectively.

TableStrDet achieves greater than 5.7% and 3.1% improvement in detecting and classifying rows and columns, respectively, on both test sets over the state-of-the-art solution. In addition to that, we show that TableStrDet has a higher generalization potential than the state-of-the-art solution. TableStrDet achieves a higher weighted average F1 score on cross dataset testing where the model is trained on tables from one type of document but tested on tables from another type of document. We have also shown that along with detecting four classes of objects, TableStrDet can generate accurate cell locations with some post-processing steps that also capture the hierarchical layouts of tables with merged cells in a single inference.

Chapter 7

Conclusion

A large volume of data is shared between global organizations on a daily basis to streamline activities and to ensure the efficient functioning of such establishments. These organizations include manufacturers, product designers, academic institutions and governments as well. They share data with each other in the form of documents such as datasheets where the critical information is stored in tables. Due to the diverse functioning of each organization the format of presenting the data in the tables is purely dependent on the authors needs to convey the information in the most concise and efficient manner. With the vast number of global organizations comes a vast number of possible formats in which the table content can be organized. However, extracting the content of these tables and assembling it for analysis can help such organizations with automation and enable them to take data driven decisions. We hypothesize that a table text extraction pipeline that can divide the entire process for datasheet (document) images into three stages. In the first stage we propose detecting the tables in these datasheet (document) images and correctly identifying all document images with tables. The second stage would then identify the rows and columns in the detected table region in order to provide context about the layout of information in the table.

In this thesis, we address the first stage of the table text extraction pipeline with our proposed a deep learning based methodology, TableDet. We show that the TableDet model can address both table detection and table image classification in datasheet (document) images in a single inference. We propose certain model enhancements such as detecting text and figures along with tables, utilizing a Cascade R-CNN architecture with deformable backbone and CIOU loss at each box head to improve the table detection potential of the TableDet model. Furthermore, we show that our proposed dual-step transfer learning

process with TAC augmented images can efficiently develop a single model to classify each object in an image as a table, text or figure.

These model and training enhancements make TableDet Model A which has a deformable backbone achieve the highest table detection performance on three public and one closed dataset. On the ICDAR 2019 (test set) it achieves the highest F1 score at 90% IOU value. On the ICDAR 2017 (test set) and Lytica (test set) Model A achieves the highest F1 score as compared to state-of-the-art table detection methods. It achieves 100% F1 score on the ICDAR 2013 dataset without using an image from this set to train or fine-tune the model. Furthermore, it utilizes its enhanced table detection capability to address the table image classification task on two public and one private dataset.

The motivation of addressing this task at this stage of the pipeline is to ensure that all images with table objects (Table) along with a lower number of images without table objects (Non-Table) are available to the next stage of the pipeline; table text extraction. In this thesis, we present the impact of utilizing a classification head which has 3 conditions to determine the label of the image based on the detection results. The TableDet Model A with the classification head is able to achieve 100% recall on three test sets. In addition to that, it achieves the highest F1 score while maintaining above 92% precision on all three test sets. This indicates that TableDet Model A would allow about 8% of Non-Table images to remain in the pipeline while promoting all Table images to the next stage. Furthermore, it can address this task without any additional training.

We propose developing table images utilizing the accurate table regions detected in the first stage of the pipeline and addressing the task of table structure detection on these images. In this thesis, we propose TableStrDet model which comprises of one Cascade R-CNN architecture to detect and classify columns as irregular columns and regular columns and another Cascade R-CNN architecture to detect and classify rows as irregular rows and regular rows. The detection performance of each Cascade R-CNN architecture is enhanced by utilizing a deformable convolution backbone and Complete IOU loss at each box head. Thus, TableStrDet surpasses the state-of-the-art table structure detection solution on the ICDAR 2013 test set and TabStructDB test set by achieving greater than 5.7% and 3.1% improvement in detecting regular rows, irregular rows, regular columns and irregular columns on both test sets over the state-of-the-art solution.

It also displays a higher generalization capability as denoted by the higher weighted average F1 score it achieves on cross dataset testing. This testing involves training the model on tables from one type of document such as administrative documents and testing it on tables from another type of document such as scientific reports. By detecting four classes of objects, TableStrDet as a post-processing step can generate accurate cell locations

in a single inference to capture the hierarchical layout of content within the tables that have merged cells. Thus, in this thesis, we present TableDet and TableStrDet, two solutions that can address table detection, table image classification and table structure detection in datasheet (document) images. Thus, in this thesis we provide solutions for stage 1 (get the table location) and stage 2 (get the cell locations) of the proposed table text extraction pipeline as shown in Fig. 1.1. At stage 3 of the proposed pipeline the cell locations from stage 2 can be used to identify the locations of text in the image and an Optical Character Recognition (OCR) model such as EasyOCR [1] can be utilized to extract the text from each cell location. While the OCR model will accurately extract the table text utilizing the cell locations from step 2 will ensure that the layout of information in the tables is also preserved.

7.1 Future Works

We achieve the highest F1 scores for table detection, table image classification and table structure detection with our presented solutions for each task. However, there are certain areas where our presented solutions can be enhanced to further improve the existing performance on these tasks.

7.1.1 Increasing the dataset size

We propose supervised deep learning based models to address the table detection and table structure detection tasks on datasheet (document) images in this thesis. To enhance the performances of these solutions we manually add location details (annotations) to each image. For the case of TableDet we manually annotated table, text and figures while for TableStrDet we manually annotated rows and columns. This is a laborious task and has a high potential for errors if the annotations are developed too quickly. Furthermore, the object detection solutions that we utilize require a large number of images upwards of 1000 images to improve their generalization capacity as shown by He et al. [143]. While we utilize the concept of transfer learning to develop these solutions, we hypothesize that training these solutions on a large datasheet (document) dataset would be more beneficial as shown by He et al. [143]. We manually annotated close to 5000 document images to train TableDet and 800 images to train TableStrDet. These documents have tables from multiple domains such as academic and manufacturing. However, there are many

[1] https://github.com/JaidedAI/EasyOCR

document images that are publicly available but do not have the annotations for tables, text and figures as well as rows and columns.

Thus, we propose a potential future work to utilize a larger datasheet (document) dataset and to avoid the error prone tasks of annotating large number of images to address these tasks. We suggest utilizing a weakly supervised object detection approach such as the method proposed by Ren et al. [144] to address these tasks. Ren et al. [144] put forward a self-training [145, 146] approach that utilizes the available labeled data and a large number of unlabeled data to train the deep learning model in a teacher-student fashion [147, 148]. Pseudo labels are generated by the model the first time it is exposed to the unlabeled data and it is then trained in an iterative manner to learn the correct object labels through self-training. The training process for this approach is much longer. However this process provides the benefit of adding more diverse images which will further help the table detection and table structure detection solutions to generalize to tables from multiple domains. Furthermore, it will reduce the time required to annotate the data which can be utilized to improve the object detection architectures to further enhance their detection performance.

A larger document image dataset can also provide an opportunity to develop a validation set along with the currently available training and test sets. By building a validation set we can also measure any potential overfitting that may appear when the models are trained. This overfitting could develop due to the similarity of document images in the dataset thus reducing the generalization capability of our proposed approach especially for cases where the proposed solution could face unseen document images. Furthermore, this additional data will provide more details on how well or how poorly our proposed solutions are able to generalize to documents that the model has not seen before.

7.1.2 Classifying table regions and identify table title and footers

In this thesis, we detect tables, text and figure regions of each datasheet (document) image. Irrespective of the presence or absence of line separators between the rows and columns of the table the TableDet model accurately detects the table and provides this table region as an input image to the TableStrDet model which then decomposes the image into rows and columns. While we achieve the highest table structure detection performance, we hypothesize, that tables without line separators or partial line separators can further benefit from detecting cells of the table rather than rows and columns as previously proposed by Prasad et al. [14]. We propose that the table detection step should include classifying the table into ruled (tables with line separators for all rows and columns) and unruled tables

(tables with partial or no line separators for all rows and columns). Thus, at the stage 2 of the proposed pipeline the table structure detection solution would be able to detect and classify rows and columns for ruled tables and cells for unruled tables. In addition we also propose identifying the table titles and footers in order to organize the text from multiple tables that our proposed solution would identify and extract from. As we already detect the text in the image we propose adding two additional object classes for our TableDet model to detect. Thus the new TableDet model would detect tables, text, figures, table title and table footer. This additional metadata would further add important tabular data that would be missed if we only detect and extract text from tables.

7.1.3 Classifying figure regions

As shown in Section 5.1 we utilize the TableDet object detection results to classify the full page images as Table or Non-Table. In our analysis for utilizing 3 conditions in the classification head of the model we inferred that figures such as charts or graphs are often mistaken for table regions by table detection solutions. As object detection solutions such as TableDet rely on the features of the object to determine the type of object, we observed that such charts share certain similarities with tables. Both types of objects are on occasion surrounded by a border line. In certain instances the line charts have vertical and horizontal lines to denote the important co-ordinates of the chart. Such line charts are often mistaken for tables as the appear with in a grid format. By classifying these figures as charts (line graph , bar graph, pie graph), maps and symbols (company logos) we hypothesize that the object detection model will learn to differentiate between such figure and table objects.

7.1.4 Unified solutions

As part of our future work we plan to combine the high precision performance of image classification models with the high recall performance of the object detection (TableDet) models. This would ensure that all Table images along with locations of the tables and much lower number of Non-Table images would be sent for further processing to the next stage of in the table text extraction pipeline; table structure detection.

For the table structure detection solution we plan to combine detection performance of the two separate Cascade R-CNN architectures of the TableStrDet model. We determined that it is more effective to utilize separate architectures in parallel where one architecture detects and classifies rows while the other detects and classifies columns. However, we

hypothesize that a single architecture with two separate RPN's and box heads but with a single feature extractor could be utilized to address this task. The feature extractor could also be enhanced to provide richer features to each of the RPN's and box heads.

References

[1] S. K. Rao and R. Prasad, "Impact of 5g technologies on industry 4.0," *Wireless Personal Communications*, vol. 100, pp. 145–159, 2018.

[2] S. Vaidya, P. Ambad, and S. Bhosle, "Industry 4.0 – a glimpse," *Procedia Manufacturing*, vol. 20, pp. 233–238, 2018. 2nd International Conference on Materials, Manufacturing and Design Engineering (iCMMD2017), 11-12 December 2017, MIT Aurangabad, Maharashtra, INDIA.

[3] B. Tjahjono, C. Esplugues, E. Ares, and G. Pelaez, "What does industry 4.0 mean to supply chain?," *Procedia Manufacturing*, vol. 13, pp. 1175 – 1182, 2017. Manufacturing Engineering Society International Conference 2017, MESIC 2017, 28-30 June 2017, Vigo (Pontevedra), Spain.

[4] R. Cerchione and E. Esposito, "A systematic review of supply chain knowledge management research: State of the art and research opportunities," *International Journal of Production Economics*, vol. 182, pp. 276 – 292, 2016.

[5] E. Brynjolfsson and K. McElheran, "Data in action: Data-driven decision making and predictive analytics in u.s. manufacturing," *Entrepreneurship & Economics eJournal*, 2019.

[6] X. Zhong, J. Tang, and A. Jimeno-Yepes, "Publaynet: Largest dataset ever for document layout analysis," *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1015–1022, 2019.

[7] A. Shigarov, A. Altaev, A. Mikhailov, V. V. Paramonov, and E. A. Cherkashin, "Tabbypdf: Web-based system for pdf table extraction," in *ICIST*, 2018.

[8] L. Gao, Y. Huang, H. Déjean, J. Meunier, Q. Yan, Y. Fang, F. Kleber, and E. Lang, "Icdar 2019 competition on table detection and recognition (ctdar)," in *2019 Interna-*

tional Conference on Document Analysis and Recognition (ICDAR), pp. 1510–1515, 2019.

[9] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016.

[10] S. Qiao, L.-C. Chen, and A. Yuille, "Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution," *ArXiv*, vol. abs/2006.02334, 2020.

[11] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6154–6162, 2018.

[12] M. R. H. Traquair, E. Kara, B. Kantarci, and S. A. Khan, "Deep learning for the detection of tabular information from electronic component datasheets," *2019 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, 2019.

[13] J. Jiang, M. Simsek, B. Kantarci, and S. A. Khan, "High precision deep learning-based tabular position detection," *2020 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, 2020.

[14] D. Prasad, A. Gadpal, K. Kapadni, M. Visave, and K. Sultanpure, "Cascadetabnet: An approach for end to end table detection and structure recognition from image-based documents," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2439–2447, 2020.

[15] S. Paliwal, D. Vishwanath, R. Rahul, M. Sharma, and L. Vig, "Tablenet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images," *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 128–133, 2019.

[16] J. Jiang, M. Simsek, B. Kantarci, and S. Khan, "Tabcellnet: Deep learning-based tabular cell structure detection," *Neurocomputing*, 2021.

[17] E. Kara, M. Traquair, M. Simsek, B. Kantarci, and S. Khan, "Holistic design for deep learning-based discovery of tabular structures in datasheet images," *Engineering Applications of Artificial Intelligence*, vol. 90, p. 103551, 2020.

[18] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-IoU loss: Faster and better learning for bounding box regression," in *AAAI*, 2020.

[19] E. Kara, "End-to-end tabular information extraction in datasheets with deep learning," 2019.

[20] R. Zanibbi, D. Blostein, and J. Cordy, "A survey of table recognition," *IJDAR*, vol. 7, pp. 1–16, 03 2004.

[21] Z. Chi, H. Huang, H.-D. Xu, H. Yu, W. Yin, and X.-L. Mao, "Complicated table structure recognition," *arXiv preprint arXiv:1908.04729*, 2019.

[22] G. Nagy, "Preliminary investigation of techniques for automated reading of unformatted text," *Commun. ACM*, vol. 11, p. 480–487, July 1968.

[23] T. Kieninger and A. Dengel, "Applying the t-recs table recognition system to the business letter domain," in *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pp. 518–522, 2001.

[24] X. Lin and Y. Xiong, "Detection and analysis of table of contents based on content association," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 8, pp. 132–143, 2005.

[25] A. Pivk, P. Cimiano, Y. Sure, M. Gams, V. Rajkovič, and R. Studer, "Transforming arbitrary tables into logical form with tartar," *Data Knowledge Engineering*, vol. 60, no. 3, pp. 567–595, 2007.

[26] Y. Liu, K. Bai, P. Mitra, and C. L. Giles, "Improving the table boundary detection in pdfs by fixing the sequence error of the sparse lines," in *2009 10th International Conference on Document Analysis and Recognition*, pp. 1006–1010, 2009.

[27] J. Fang, L. Gao, K. Bai, R. Qiu, X. Tao, and Z. Tang, "A table detection method for multipage pdf documents via visual seperators and tabular structures," in *2011 International Conference on Document Analysis and Recognition*, pp. 779–783, 2011.

[28] Y. Liu, K. Bai, P. Mitra, and C. L. Giles, "Tableseer: Automatic table metadata extraction and searching in digital libraries," in *In Technical Report*, pp. 91–100, 2007.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[30] F. M. Wahl, K. Y. Wong, and R. G. Casey, "Block segmentation and text extraction in mixed text/image documents," *Computer Graphics and Image Processing*, vol. 20, no. 4, pp. 375–390, 1982.

[31] A. Jain and B. Yu, "Document representation and its application to page decomposition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 294–308, 1998.

[32] D. N. Tran, T. A. P. Tran, A. Oh, S.-H. Kim, and I. S. Na, "Table detection from document image using vertical arrangement of text blocks," *International Journal of Contents*, vol. 11, pp. 77–85, 2015.

[33] B. Gatos, D. Danatsas, I. Pratikakis, and S. Perantonis, "Automatic table detection in document images," vol. 3686, pp. 609–618, 08 2005.

[34] F. Cesarini, S. Marinai, L. Sarti, and G. Soda, "Trainable table location in document images," in *2002 International Conference on Pattern Recognition*, vol. 3, pp. 236–240 vol.3, 2002.

[35] T. Kieninger and A. Dengel, "The t-recs table recognition and analysis system," in *Document Analysis Systems: Theory and Practice* (S.-W. Lee and Y. Nakano, eds.), (Berlin, Heidelberg), pp. 255–270, Springer Berlin Heidelberg, 1999.

[36] Y. Wang, R. Haralick, and I. Phillips, "Zone content classification and its performance evaluation," in *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pp. 540–544, 2001.

[37] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[38] Y. Wang and J. Hu, "A machine learning based approach for table detection on the web," 10 2002.

[39] D. Tkaczyk, P. Szostek, P. Dendek, M. Fedoryszak, and Bolikowski, "Cermine – automatic extraction of metadata and references from scientific literature," 04 2014.

[40] H. Ng, C. Y. Lim, and J. L. T. Koo, "Learning to recognize tables in free text," in *ACL*, 1999.

[41] T. Kasar, P. Barlas, S. Adam, C. Chatelain, and T. Paquet, "Learning to detect tables in scanned document images using line information," in *2013 12th International Conference on Document Analysis and Recognition*, pp. 1185–1189, 2013.

[42] M. Fan and D. S. Kim, "Table region detection on large-scale PDF files without labeled data," *CoRR*, vol. abs/1506.08891, 2015.

[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[45] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, 2015.

[46] A. Bochkovskiy, C.-Y. Wang, and H. Liao, "Yolov4: Optimal speed and accuracy of object detection," *ArXiv*, vol. abs/2004.10934, 2020.

[47] S. Wei, X. Zeng, Q. Qu, M. Wang, H. Su, and J. Shi, "Hrsid: A high-resolution sar images dataset for ship detection and instance segmentation," *IEEE Access*, vol. 8, pp. 120234–120254, 2020.

[48] Q. Lin, Y. Ding, H. Xu, W. Lin, J. Li, and X. Xie, "Ecascade-rcnn: Enhanced cascade rcnn for multi-scale object detection in uav images," in *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*, pp. 268–272, 2021.

[49] L. Hao, L. Gao, X. Yi, and Z. Tang, "A table detection method for pdf documents based on convolutional neural networks," in *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pp. 287–292, 2016.

[50] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed, "Deepdesrt: Deep learning for detection and structure recognition of tables in document images," *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, pp. 1162–1167, 2017.

[51] S. A. Siddiqui, P. I. Khan, A. Dengel, and S. Ahmed, "Rethinking semantic segmentation for table structure recognition in documents," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1397–1402, 2019.

[52] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask r-cnn," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

[53] E. Kara, M. Traquair, B. Kantarci, and S. Khan, "Deep learning for recognizing the anatomy of tables on datasheets," in *2019 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, 2019.

[54] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995, 2017.

[55] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 764–773, 2017.

[56] S. A. Siddiqui, I. A. Fateh, S. T. R. Rizvi, A. Dengel, and S. Ahmed, "Deeptabstr: Deep learning based table structure recognition," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1403–1409, 2019.

[57] N. Sun, Y. Zhu, and X. Hu, "Faster r-cnn based table detection combining corner locating," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1314–1319, 2019.

[58] X.-H. Li, F. Yin, and C.-L. Liu, "Page object detection from pdf document images by deep structured prediction and supervised clustering," in *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 3627–3632, 2018.

[59] I. Kavasidis, S. Palazzo, C. Spampinato, C. Pino, D. Giordano, D. Giuffrida, and P. Messina, "A saliency-based convolutional neural network for table and chart detection in digitized documents," in *ICIAP*, 2019.

[60] Y. Li, L. Gao, Z. Tang, Q. Yan, and Y. Huang, "A gan-based feature generator for table detection," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 763–768, 2019.

[61] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014.

[62] S. Arif and F. Shafait, "Table detection in document images using foreground and background features," in *2018 Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1–8, 2018.

[63] P. Pyreddy and W. B. Croft, "Tintin: A system for retrieval in text tables," in *Proceedings of the Second ACM International Conference on Digital Libraries*, DL '97, (New York, NY, USA), p. 193–200, Association for Computing Machinery, 1997.

[64] F. Cesarini, M. Gori, S. Marinai, and G. Soda, "Structured document segmentation and representation by the modified x-y tree," *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318)*, pp. 563–566, 1999.

[65] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak, "Towards domain-independent information extraction from web tables," in *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, (New York, NY, USA), p. 71–80, Association for Computing Machinery, 2007.

[66] E. Oro and M. Ruffolo, "Pdf-trex: An approach for recognizing and extracting tables from pdf documents," in *2009 10th International Conference on Document Analysis and Recognition*, pp. 906–910, 2009.

[67] B. Yildiz, K. Kaiser, and S. Miksch, "pdf2table: A method to extract table information from pdf files.," pp. 1773–1785, 01 2005.

[68] M. Göbel, T. Hassan, E. Oro, and G. Orsi, "Icdar 2013 table competition," in *2013 12th International Conference on Document Analysis and Recognition*, pp. 1449–1453, 2013.

[69] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 2004.

[70] L. Rabiner and B. Juang, "An introduction to hidden markov models," *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[71] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[72] G. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[73] S. Shetty, H. Srinivasan, M. J. Beal, and S. Srihari, "Segmentation and labeling of documents using conditional random fields," in *Electronic Imaging*, 2007.

[74] A. Quattoni, M. Collins, and T. Darrell, "Conditional random fields for object recognition," in *NIPS*, 2004.

[75] J. Lafferty, A. Mccallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," pp. 282–289, 01 2001.

[76] A. C. e. Silva, "Learning rich hidden markov models in document analysis: Table location," in *2009 10th International Conference on Document Analysis and Recognition*, pp. 843–847, 2009.

[77] M. Fan, D. Zhao, Q. Zhou, Z. Liu, T. F. Zheng, and E. Y. Chang, "Distant supervision for relation extraction with matrix completion," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Baltimore, Maryland), pp. 839–849, Association for Computational Linguistics, June 2014.

[78] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 640–651, 2017.

[79] L. Gao, X. Yi, Z. Jiang, L. Hao, and Z. Tang, "Icdar2017 competition on page object detection," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, pp. 1417–1422, 2017.

[80] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *International Conference on Learning Representations (ICLR)*, vol. abs/1511.07122, 2016.

[81] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang, "Bounding box regression with uncertainty for accurate object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[82] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "Hybrid task cascade for instance segmentation," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4969–4978, 2019.

[83] Y. Liu, Y. Wang, S. Wang, T. Liang, Q. Zhao, Z. Tang, and H. Ling, "Cbnet: A novel composite backbone network architecture for object detection," in *AAAI*, 2020.

[84] N. Bodla, B. Singh, R. Chellappa, and L. Davis, "Soft-nms — improving object detection with one line of code," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5562–5570, 2017.

[85] S. A. Siddiqui, M. I. Malik, S. Agne, A. Dengel, and S. Ahmed, "Decnt: Deep deformable cnn for table detection," *IEEE Access*, vol. 6, pp. 74151–74161, 2018.

[86] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2017.

[87] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[88] S. A. Khan, S. Khalid, M. Shahzad, and F. Shafait, "Table structure extraction with bi-directional gated recurrent unit networks," pp. 1366–1371, 09 2019.

[89] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[90] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[91] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[92] Y. Huang, Q. Yan, Y. Li, Y. Chen, X. Wang, L. Gao, and Z. Tang, "A yolo-based table detection method," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 813–818, 2019.

[93] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, "Deep high-resolution representation learning for

visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[94] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[95] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, pp. 807–814, 2010.

[96] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[97] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[98] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick, "Momentum contrast for unsupervised visual representation learning," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9726–9735, 2020.

[99] X. Chen, L. Xie, J. Wu, and Q. Tian, "Cyclic cnn: Image classification with multiscale and multilocation contexts," *IEEE Internet of Things Journal*, vol. 8, no. 9, pp. 7466–7475, 2021.

[100] Y. Pei, Y. Huang, Q. Zou, X. Zhang, and S. Wang, "Effects of image degradation and degradation removal to cnn-based image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, pp. 1239–1253, 2021.

[101] Z. KARHAN and F. AKAL, "Covid-19 classification using deep learning in chest x-ray images," in *2020 Medical Technologies Congress (TIPTEKNO)*, pp. 1–4, 2020.

[102] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision*, pp. 213–229, Springer, 2020.

[103] C. Zhou, J. Song, S. Zhou, Z. Zhang, and J. Xing, "Covid-19 detection based on image regrouping and resnet-svm using chest x-ray images," *IEEE Access*, vol. 9, pp. 81902–81912, 2021.

[104] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9404–9413, 2019.

[105] J. Liu, Y. Gu, S. Han, Z. Zhang, J. Guo, and X. Cheng, "Feature rescaling and fusion for tiny object detection," *IEEE Access*, vol. 9, pp. 62946–62955, 2021.

[106] G. Kalyani, B. Janakiramaiah, L. Prasad, A. Karuna, and A. M. Babu, "Efficient crowd counting model using feature pyramid network and resnext," *Soft Comput.*, vol. 25, pp. 10497–10507, 2021.

[107] Z. Zhou, Y. Li, C. Peng, H. Wang, and S. Du, "Image processing: Facilitating retinanet for detecting small objects," *Journal of Physics: Conference Series*, vol. 1815, 2021.

[108] L. Yang, Q. Song, Z. Wang, M. Hu, and C. Liu, "Hier r-cnn: Instance-level human parts detection and a new benchmark," *IEEE Transactions on Image Processing*, vol. 30, pp. 39–54, 2021.

[109] L. Zhang, D. Dong, Z. Liu, J. Zhou, and J. Tian, "Joint multi-task learning for survival prediction of gastric cancer patients using ct images," in *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pp. 895–898, 2021.

[110] H. Peng and S. Yu, "A systematic iou-related method: Beyond simplified regression for better localization," *IEEE Transactions on Image Processing*, vol. 30, pp. 5032–5044, 2021.

[111] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[112] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[113] X. Lu, Q. Li, B. Li, and J. Yan, "Mimicdet: Bridging the gap between one-stage and two-stage object detection," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pp. 541–557, Springer, 2020.

[114] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[115] R. B. Girshick, "Fast r-cnn," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

[116] A. Groener, G. Chern, and M. Pritt, "A comparison of deep learning object detection models for satellite imagery," in *2019 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pp. 1–10, 2019.

[117] S. Xu, H. Lu, M. Ye, K. Yan, W. Zhu, and Q. Jin, "Improved cascade r-cnn for medical images of pulmonary nodules detection combining dilated hrnet," (New York, NY, USA), Association for Computing Machinery, 2020.

[118] W. Zhao, H. Huang, D. Li, F. Chen, and W. Cheng, "Pointer defect detection based on transfer learning and improved cascade-rcnn," *Sensors*, vol. 20, no. 17, 2020.

[119] H. Zhao and S. Ram, "Constrained cascade generalization of decision trees," *IEEE Transactions on Knowledge and Data Engineering*, p. 738, 2004.

[120] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. S. Huang, "Unitbox: An advanced object detection network," *CoRR*, vol. abs/1608.01471, 2016.

[121] S. H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. D. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 658–666, 2019.

[122] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *International Conference on Learning Representations, ICLR 2017*, 2017.

[123] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, p. III–1058–III–1066, JMLR.org, 2013.

[124] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proceedings of the British Machine Vision Conference (BMVC)* (E. R. H. Richard C. Wilson and W. A. P. Smith, eds.), pp. 87.1–87.12, BMVA Press, September 2016.

[125] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[126] T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *CoRR*, vol. abs/1708.04552, 2017.

[127] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Dropblock: A regularization method for convolutional networks," in *NeurIPS*, 2018.

[128] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *AAAI*, 2020.

[129] R. Fong, "Occlusions for effective data augmentation in image classification," *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, Oct 2019.

[130] J. Gilmer, N. Ford, N. Carlini, and E. Cubuk, "Adversarial examples are a natural consequence of test error in noise," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 2280–2289, PMLR, 09–15 Jun 2019.

[131] D. Yin, R. G. Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer, *A Fourier Perspective on Model Robustness in Computer Vision*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[132] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[133] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, "Fast autoaugment," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[134] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V. Le, "Learning data augmentation strategies for object detection," in *European Conference on Computer Vision*, pp. 566–583, Springer, 2020.

[135] R. Rastan, "Towards generic framework for tabular data extraction and management in documents," in *PIKM '13*, 2013.

[136] J. Yi, H. Tang, P. Wu, B. Liu, D. J. Hoeppner, D. N. Metaxas, L. Han, and W. Fan, "Object-guided instance segmentation for biological images.," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 12677–12684, 2020.

[137] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li, "Solo: Segmenting objects by locations," in *European Conference on Computer Vision*, pp. 649–665, Springer, 2020.

[138] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[139] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[140] Y. Wu and K. He, "Group normalization," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

[141] X. Yang, E. Yumer, P. Asente, M. Kraley, D. Kifer, and C. L. Giles, "Learning to extract semantic structure from documents using multimodal fully convolutional neural networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4342–4351, 2017.

[142] M. Hansen, A. Pomp, K. Erki, and T. Meisen, "Data-driven recognition and extraction of pdf document elements," *Technologies*, vol. 7, p. 65, 09 2019.

[143] K. He, R. Girshick, and P. Dollar, "Rethinking imagenet pre-training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[144] Z. Ren, Z. Yu, X. Yang, M.-Y. Liu, Y. J. Lee, A. Schwing, and J. Kautz, "Instance-aware, context-focused, and memory-efficient weakly supervised object detection," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10595–10604, 2020.

[145] Y. Zou, Z. Yu, X. Liu, B. V. Kumar, and J. Wang, "Confidence regularized self-training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[146] Y. Zou, Z. Yu, B. Kumar, and J. Wang, "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training," in *ECCV*, 2018.

[147] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10687–10698, 2020.

[148] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. Le, "Rethinking pre-training and self-training," in *NeurIPS*, 2020.

Appendix A

Appendix

A.1 Table Detection

We propose TableDet, a deep learning based approach to detect table, text and figures in datasheet (document) images. The TableDet model utilizes a deformabl convolution backbone and CIOU loss to improve it's table detection performance. We train the model in a dual-step transfer learning process with Table Aware Cutout augmented images to further enhance it's table detection performance. In Section A.1.1 we describe the public and closed datasets utilized to train the TableDet model and In Section A.1.2 we display the inference results of the TableDet Model A.

A.1.1 Datasets

We utilize four public and one closed dataset to develop the public and private datasets as shown in Table 4.1. The TableDet model variants are trained on the public and private datasets in order to fairly compare the table detection performance improvements with state-of-the-art solutions that utilize similar datasets. In the following sections we describe each dataset in detail.

ICDAR 2013

The International Conference on Document Analysis and Recognition (ICDAR) is held every two years to display the latest works in the field of document analysis. In 2013 the

conference held a competition for two tasks; table detection and table structure detection [68]. The competition provided a dataset of publicly available documents from European Union and United State of America as a benchmark on which different solutions could try and solve these two tasks. The dataset comprises of 67 documents with a total of 238 pages. In this work, we have converted each page into an image. It is also worth noting that the 238 images are only used for testing and not a single image is used to train either Model A or Model B variants of the TableDet model as shown in Table 4.1.

ICDAR 2017

The ICDAR 2017 Page Object Detection competition [79] is proposed as a document layout analysis competition to identify multiple object types in document images. Unlike ICDAR 2013 [68] Table competition, the POD competition was setup up for document analysis solutions to detect tables, and other objects such as figures in the image. The dataset consists of 2417 images. Each image is a page from scientific reports available on CiteSeer. We continue to use the train and test splits setup by the competition in this work to fairly train and test our work with other state-of-the-art methods on table detection. Thus, as shown in Table 4.1 we have utilized 1600 images for training and 817 images for testing.

ICDAR 2019

The ICDAR 2019 Competition on Table Detection and Recognition [8] provided two types of datasets for table detection. One subset contained modern document pages while the other contained archival document pages. These pages are provided as images along with the annotations for table locations. In this work we have only used the modern document images which comprises of product and scientific reports. In addition to that, the images are a mixture of Chinese and English document pages converted to images. As shown in Table 4.1 we have maintained the same train and test splits as mentioned in [8]. 600 images are used to train the model while 240 images are used for testing.

Marmot

The Marmot dataset was developed by Peking University Computer Science Department. It is a widely recognized dataset for the task of table detection used in previous works such as [12, 13]. It consists of 2000 document images of varied topics including academic papers. 1000 images are Chinese pages from e-books and have a one-column layout.

While the other 1000 images are English academic papers available on CiteSeer. Similar to the other datasets we added the annotations of figures and text to the existing table annotations for all 2000 images. We used the entire 2000 images for training in this work.

Lytica

In addition to using the public datasets such as ICDAR 13 we have also used a closed dataset provided by our industry partner Lytica Inc. that specializes in supply chain optimization and data analysis. We refer to this dataset as the Lytica dataset. It consists of 1240 images of datasheet pages converted into images from various supply chain organizations. These datasheet images contain more heterogeneous tables than the public datasets. We define heterogeneous tables as tables where row and columns share a merged cell to display data with a hierarchical layout [16]. This dataset also contains tables which have smaller tables within them. We utilized 1000 images for training and 240 images for testing as shown in Table 4.1.

A.1.2 Results

We display the detection capability of the TableDet model by highlighting detected tables in red, text in blue and figures in green colored boxes in the sample images. As shown in Fig. A.1, Fig. A.3, Fig. A.4 and Fig. A.5 the vertically orientated tables along with figure and text are accurately detected. There are overlapping bounding boxes for the text regions which we hope to resolve as part of our future work. Similarly there are overlapping text boxes within the figure images and vice versa which we also hope to resolve as part of our future work. In Fig. A.2 and Fig. A.6 we display the accurate detection capability of TableDet to detect horizontally oriented tables as well. It is worth noting that irrespective of the scale of the image, the presence or absence of line separators between cells, the presence or absence of merged cells all tables are accurately detected by the TableDet model on images of all four test sets.

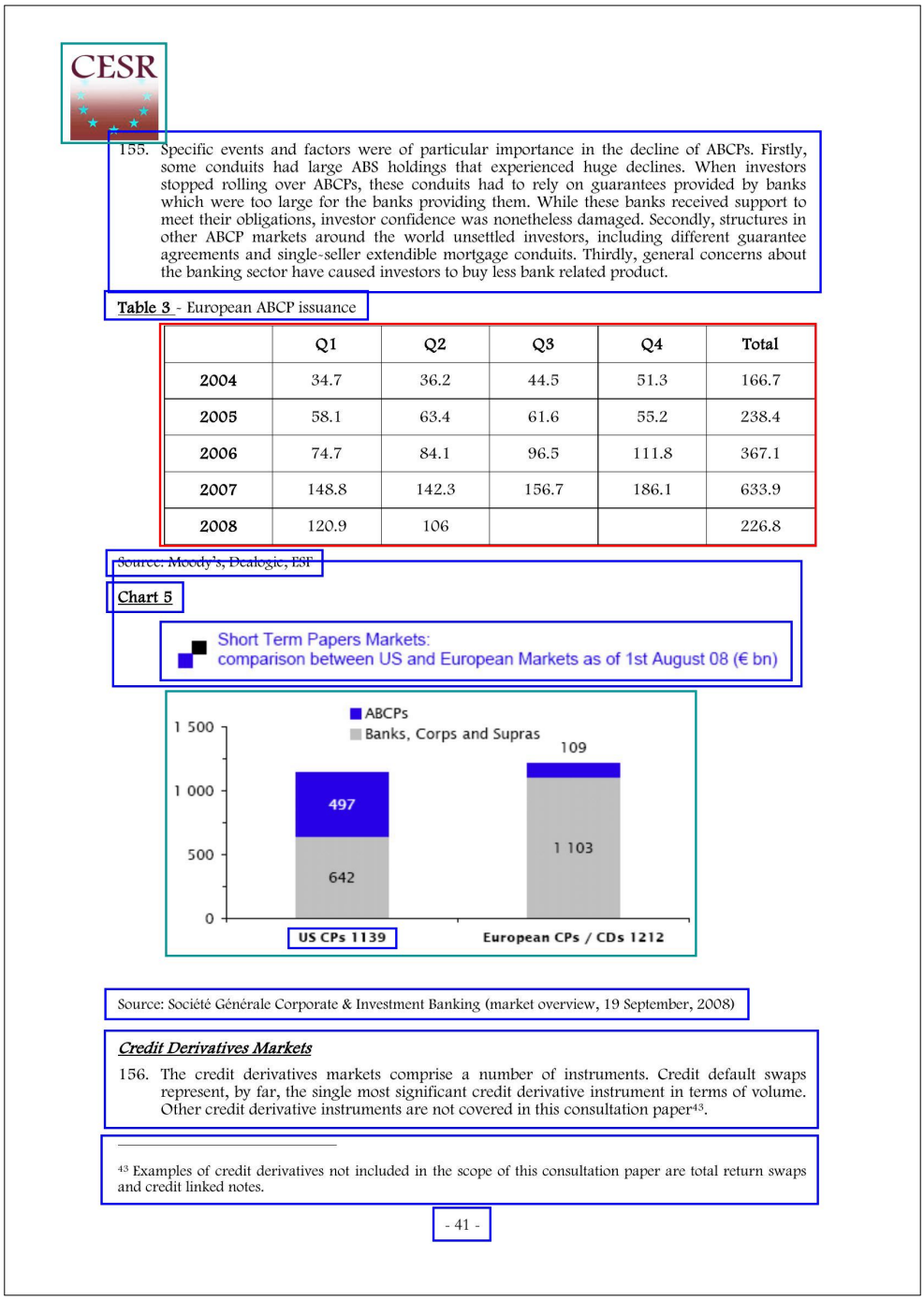


Figure A.1: **TableDet inference result on sample image from ICDAR 2013 dataset**

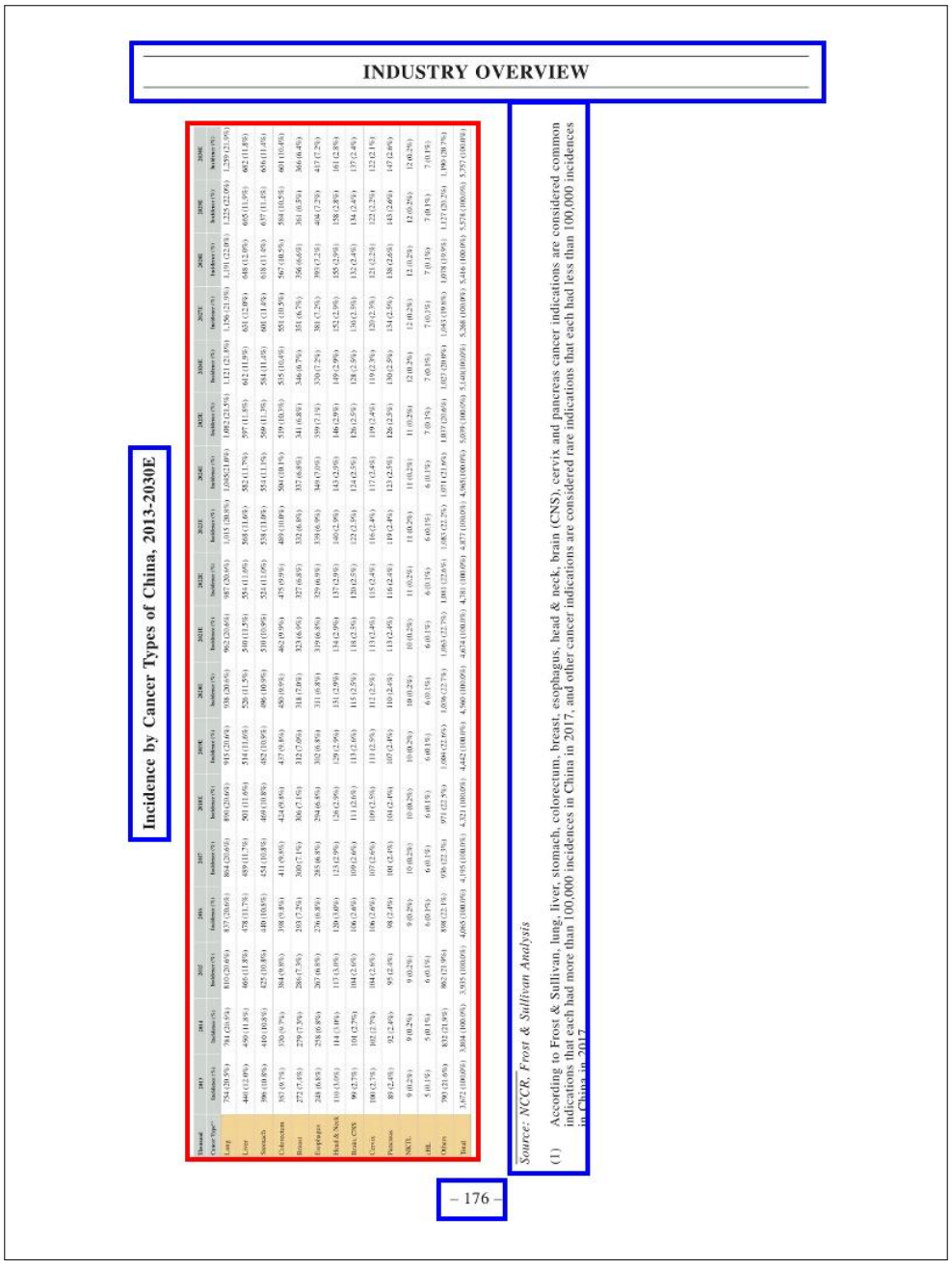


Figure A.2: **TableDet inference result on sample image from ICDAR 2019 test set**

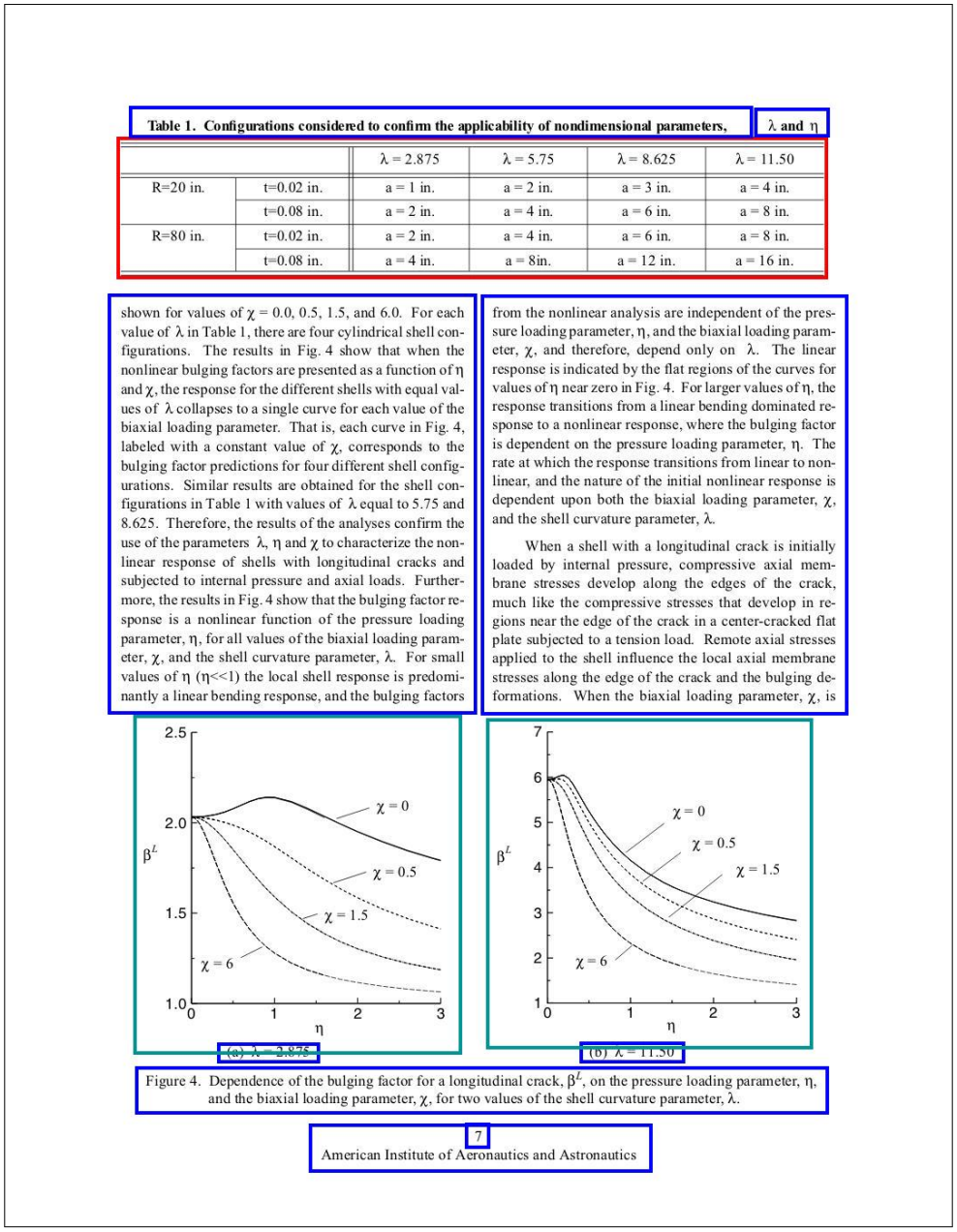


Table 1. Configurations considered to confirm the applicability of nondimensional parameters, λ and η

		λ = 2.875	λ = 5.75	λ = 8.625	λ = 11.50
R=20 in.	t=0.02 in.	a = 1 in.	a = 2 in.	a = 3 in.	a = 4 in.
	t=0.08 in.	a = 2 in.	a = 4 in.	a = 6 in.	a = 8 in.
R=80 in.	t=0.02 in.	a = 2 in.	a = 4 in.	a = 6 in.	a = 8 in.
	t=0.08 in.	a = 4 in.	a = 8in.	a = 12 in.	a = 16 in.

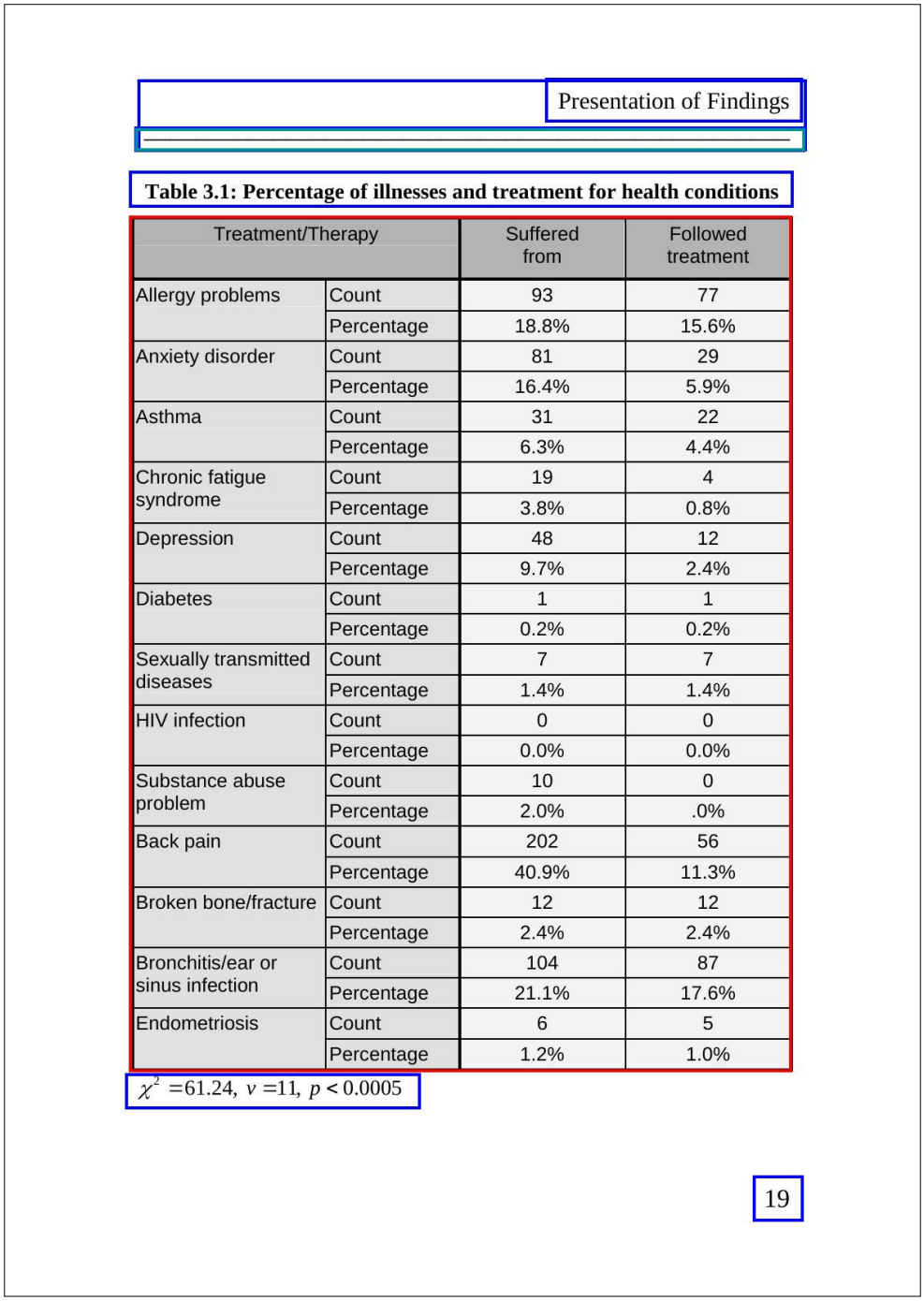
shown for values of χ = 0.0, 0.5, 1.5, and 6.0. For each value of λ in Table 1, there are four cylindrical shell configurations. The results in Fig. 4 show that when the nonlinear bulging factors are presented as a function of η and χ, the response for the different shells with equal values of λ collapses to a single curve for each value of the biaxial loading parameter. That is, each curve in Fig. 4, labeled with a constant value of χ, corresponds to the bulging factor predictions for four different shell configurations. Similar results are obtained for the shell configurations in Table 1 with values of λ equal to 5.75 and 8.625. Therefore, the results of the analyses confirm the use of the parameters λ, η and χ to characterize the nonlinear response of shells with longitudinal cracks and subjected to internal pressure and axial loads. Furthermore, the results in Fig. 4 show that the bulging factor response is a nonlinear function of the pressure loading parameter, η, for all values of the biaxial loading parameter, χ, and the shell curvature parameter, λ. For small values of η (η<<1) the local shell response is predominantly a linear bending response, and the bulging factors from the nonlinear analysis are independent of the pressure loading parameter, η, and the biaxial loading parameter, χ, and therefore, depend only on λ. The linear response is indicated by the flat regions of the curves for values of η near zero in Fig. 4. For larger values of η, the response transitions from a linear bending dominated response to a nonlinear response, where the bulging factor is dependent on the pressure loading parameter, η. The rate at which the response transitions from linear to nonlinear, and the nature of the initial nonlinear response is dependent upon both the biaxial loading parameter, χ, and the shell curvature parameter, λ.

When a shell with a longitudinal crack is initially loaded by internal pressure, compressive axial membrane stresses develop along the edges of the crack, much like the compressive stresses that develop in regions near the edge of the crack in a center-cracked flat plate subjected to a tension load. Remote axial stresses applied to the shell influence the local axial membrane stresses along the edge of the crack and the bulging deformations. When the biaxial loading parameter, χ, is

(a) λ = 2.875 (b) λ = 11.50

Figure 4. Dependence of the bulging factor for a longitudinal crack, β^L, on the pressure loading parameter, η, and the biaxial loading parameter, χ, for two values of the shell curvature parameter, λ.

Figure A.3: **TableDet inference result on sample image from ICDAR 2018 test set**



Presentation of Findings

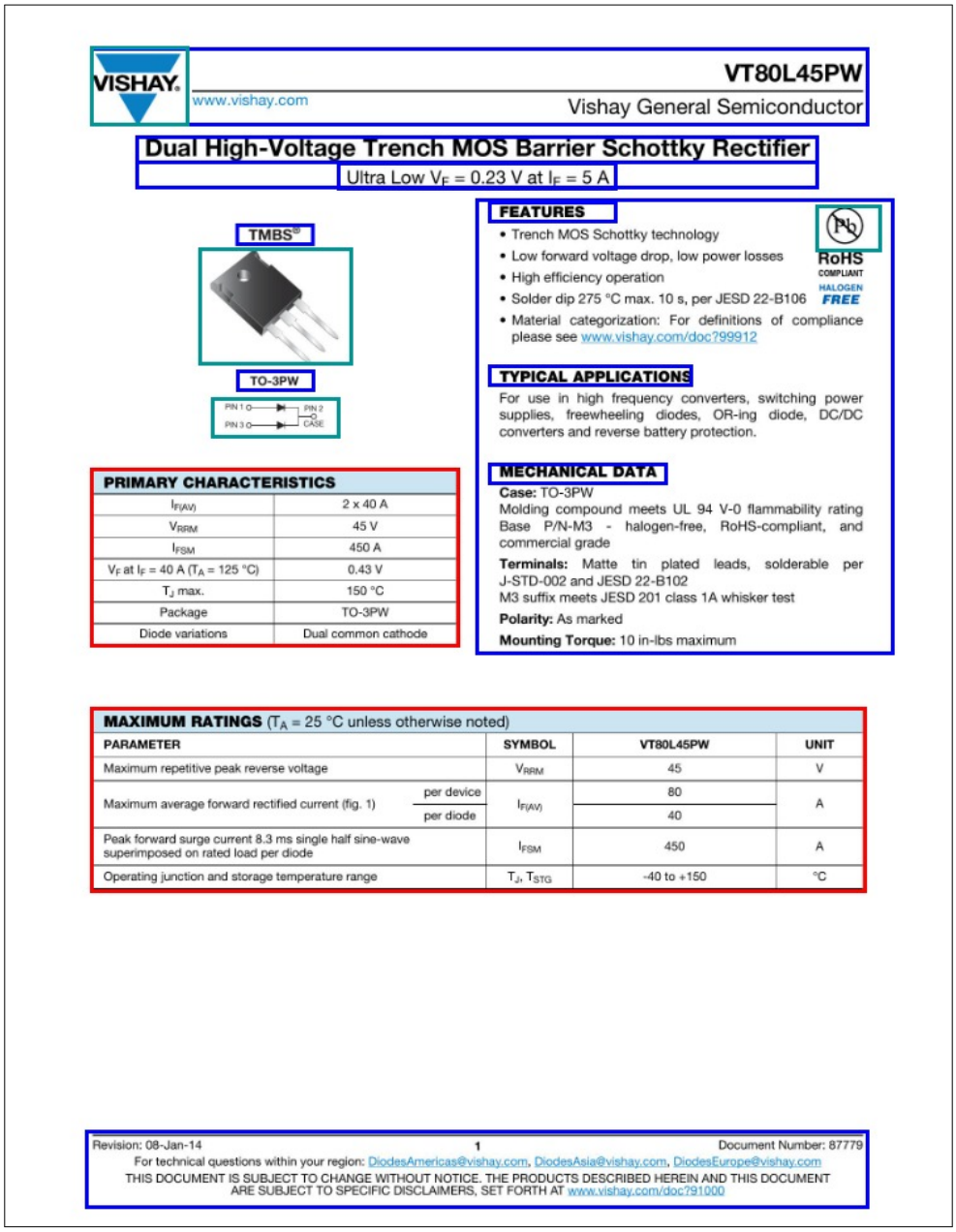
Table 3.1: Percentage of illnesses and treatment for health conditions

Treatment/Therapy		Suffered from	Followed treatment
Allergy problems	Count	93	77
	Percentage	18.8%	15.6%
Anxiety disorder	Count	81	29
	Percentage	16.4%	5.9%
Asthma	Count	31	22
	Percentage	6.3%	4.4%
Chronic fatigue syndrome	Count	19	4
	Percentage	3.8%	0.8%
Depression	Count	48	12
	Percentage	9.7%	2.4%
Diabetes	Count	1	1
	Percentage	0.2%	0.2%
Sexually transmitted diseases	Count	7	7
	Percentage	1.4%	1.4%
HIV infection	Count	0	0
	Percentage	0.0%	0.0%
Substance abuse problem	Count	10	0
	Percentage	2.0%	.0%
Back pain	Count	202	56
	Percentage	40.9%	11.3%
Broken bone/fracture	Count	12	12
	Percentage	2.4%	2.4%
Bronchitis/ear or sinus infection	Count	104	87
	Percentage	21.1%	17.6%
Endometriosis	Count	6	5
	Percentage	1.2%	1.0%

$\chi^2 = 61.24,\ v = 11,\ p < 0.0005$

19

Figure A.4: **TableDet inference result on sample image from ICDAR 2013 dataset**



VISHAY
www.vishay.com

VT80L45PW
Vishay General Semiconductor

Dual High-Voltage Trench MOS Barrier Schottky Rectifier

Ultra Low V_F = 0.23 V at I_F = 5 A

TMBS®

TO-3PW

PIN 1 PIN 2 PIN 3 CASE

FEATURES

- Trench MOS Schottky technology
- Low forward voltage drop, low power losses
- High efficiency operation
- Solder dip 275 °C max. 10 s, per JESD 22-B106
- Material categorization: For definitions of compliance please see www.vishay.com/doc?99912

RoHS COMPLIANT
HALOGEN FREE

TYPICAL APPLICATIONS

For use in high frequency converters, switching power supplies, freewheeling diodes, OR-ing diode, DC/DC converters and reverse battery protection.

MECHANICAL DATA

Case: TO-3PW
Molding compound meets UL 94 V-0 flammability rating Base P/N-M3 - halogen-free, RoHS-compliant, and commercial grade

Terminals: Matte tin plated leads, solderable per J-STD-002 and JESD 22-B102
M3 suffix meets JESD 201 class 1A whisker test

Polarity: As marked

Mounting Torque: 10 in-lbs maximum

PRIMARY CHARACTERISTICS

$I_{F(AV)}$	2 x 40 A
V_{RRM}	45 V
I_{FSM}	450 A
V_F at I_F = 40 A (T_A = 125 °C)	0.43 V
T_J max.	150 °C
Package	TO-3PW
Diode variations	Dual common cathode

MAXIMUM RATINGS (T_A = 25 °C unless otherwise noted)

PARAMETER		SYMBOL	VT80L45PW	UNIT
Maximum repetitive peak reverse voltage		V_{RRM}	45	V
Maximum average forward rectified current (fig. 1)	per device	$I_{F(AV)}$	80	A
	per diode		40	
Peak forward surge current 8.3 ms single half sine-wave superimposed on rated load per diode		I_{FSM}	450	A
Operating junction and storage temperature range		T_J, T_{STG}	-40 to +150	°C

Figure A.5: **TableDet inference result on sample image from Lytica test set**

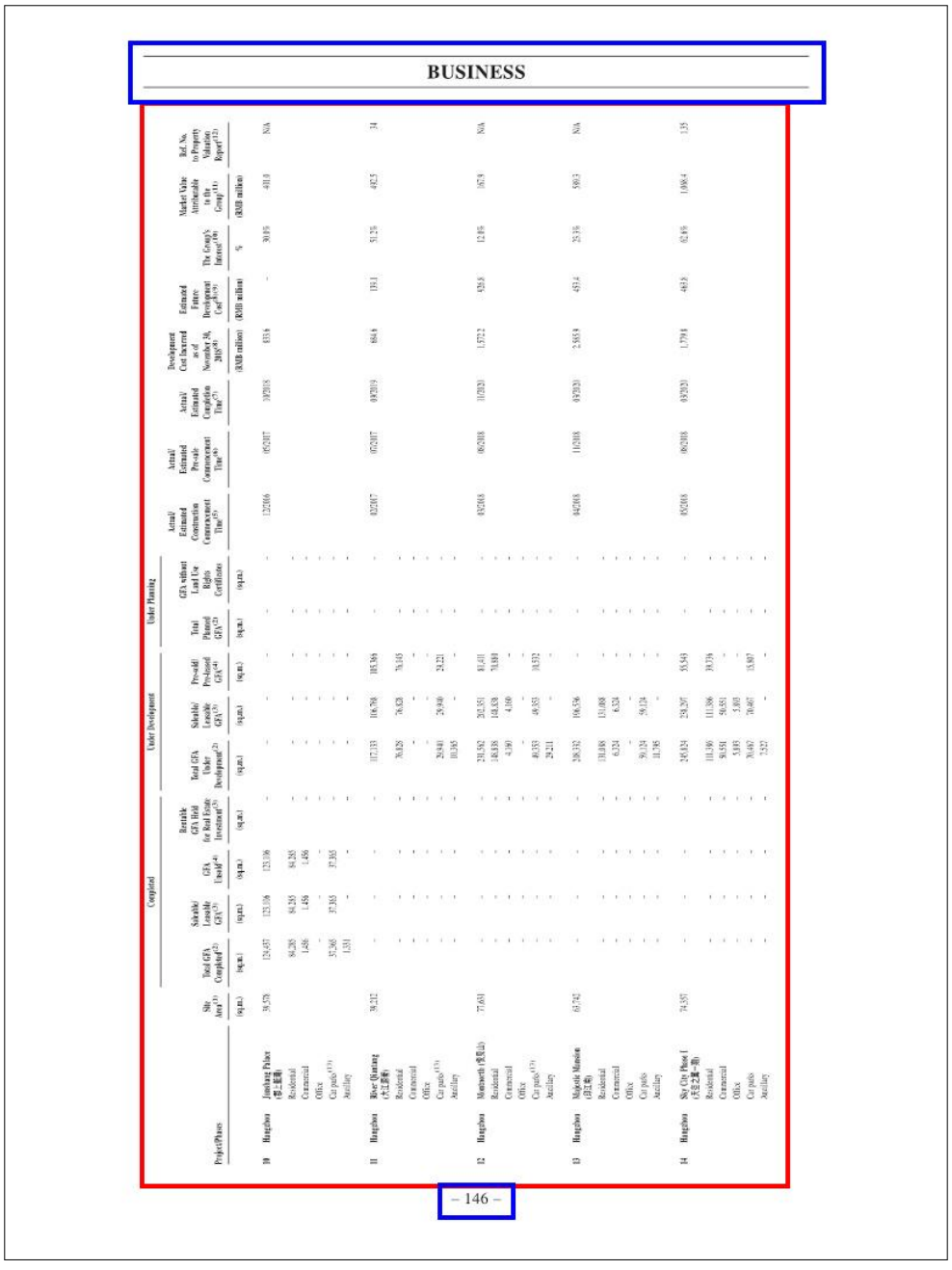
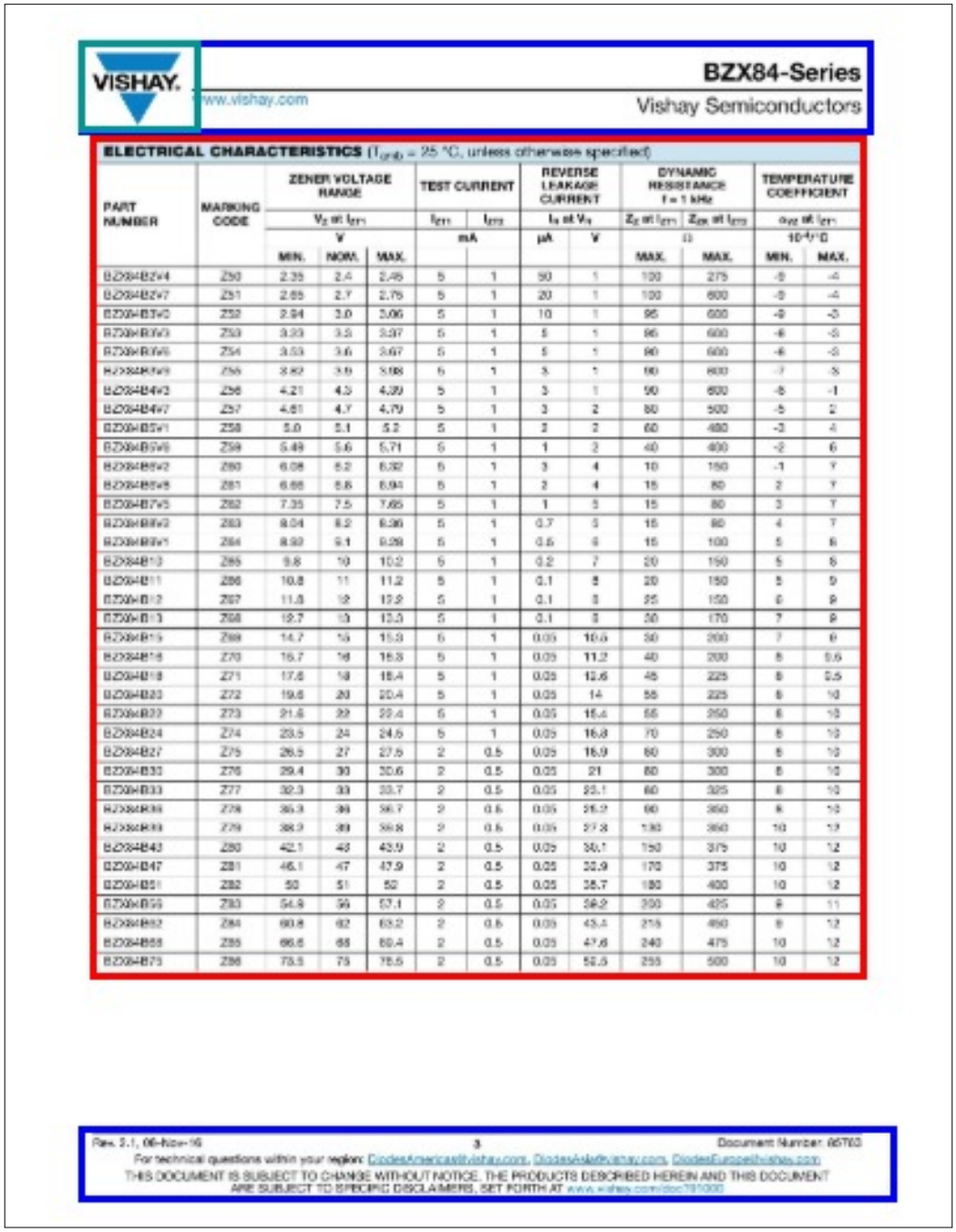


Figure A.6: **TableDet inference result on sample image from ICDAR 2019 test set**



VISHAY. www.vishay.com

BZX84-Series

Vishay Semiconductors

ELECTRICAL CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)

PART NUMBER	MARKING CODE	ZENER VOLTAGE RANGE V_Z at I_{ZT1}			TEST CURRENT I_{ZT1}	TEST CURRENT I_{ZT2}	REVERSE LEAKAGE CURRENT I_R at V_R		DYNAMIC RESISTANCE f = 1 kHz Z_Z at I_{ZT1}	DYNAMIC RESISTANCE f = 1 kHz Z_{ZK} at I_{ZT2}	TEMPERATURE COEFFICIENT α_{VZ} at I_{ZT1}	TEMPERATURE COEFFICIENT α_{VZ} at I_{ZT1}
		V			mA		µA	V	Ω		10^{-4}/°C	
		MIN.	NOM.	MAX.					MAX.	MAX.	MIN.	MAX.
BZX84Z4V4	Z50	2.35	2.4	2.45	5	1	50	1	100	275	-9	-4
BZX84Z4V7	Z51	2.65	2.7	2.75	5	1	20	1	100	600	-9	-4
BZX84Z5V0	Z52	2.94	3.0	3.06	5	1	10	1	95	600	-9	-5
BZX84Z5V3	Z53	3.23	3.3	3.37	5	1	5	1	95	600	-8	-5
BZX84Z5V6	Z54	3.53	3.6	3.67	5	1	5	1	90	600	-8	-3
BZX84Z6V2	Z55	3.82	3.9	3.98	5	1	3	1	90	600	-7	-3
BZX84B4V3	Z56	4.21	4.3	4.39	5	1	3	1	90	600	-6	-1
BZX84B4V7	Z57	4.61	4.7	4.79	5	1	3	2	80	500	-5	2
BZX84B5V1	Z58	5.0	5.1	5.2	5	1	2	2	60	480	-3	4
BZX84B5V6	Z59	5.49	5.6	5.71	5	1	1	2	40	400	-2	6
BZX84B6V2	Z60	6.08	6.2	6.32	5	1	3	4	10	150	-1	7
BZX84B6V8	Z61	6.66	6.8	6.94	5	1	2	4	15	80	2	7
BZX84B7V5	Z62	7.35	7.5	7.65	5	1	1	5	15	80	3	7
BZX84B8V2	Z63	8.04	8.2	8.36	5	1	0.7	5	15	80	4	7
BZX84B9V1	Z64	8.93	9.1	9.28	5	1	0.5	6	15	100	5	8
BZX84C10	Z65	9.8	10	10.2	5	1	0.2	7	20	150	5	8
BZX84C11	Z66	10.8	11	11.2	5	1	0.1	8	20	150	5	9
BZX84C12	Z67	11.8	12	12.2	5	1	0.1	8	25	150	6	9
BZX84C13	Z68	12.7	13	13.3	5	1	0.1	8	30	170	7	9
BZX84C15	Z69	14.7	15	15.3	5	1	0.05	10.5	30	200	7	9
BZX84C16	Z70	15.7	16	16.3	5	1	0.05	11.2	40	200	8	9.5
BZX84C18	Z71	17.6	18	18.4	5	1	0.05	12.6	45	225	8	9.5
BZX84C20	Z72	19.6	20	20.4	5	1	0.05	14	55	225	8	10
BZX84C22	Z73	21.6	22	22.4	5	1	0.05	15.4	55	250	8	10
BZX84C24	Z74	23.5	24	24.5	5	1	0.05	16.8	70	250	8	10
BZX84C27	Z75	26.5	27	27.5	2	0.5	0.05	18.9	80	300	8	10
BZX84C30	Z76	29.4	30	30.6	2	0.5	0.05	21	80	300	8	10
BZX84C33	Z77	32.3	33	33.7	2	0.5	0.05	23.1	80	325	8	10
BZX84C36	Z78	35.3	36	36.7	2	0.5	0.05	25.2	90	350	8	10
BZX84C39	Z79	38.3	39	39.8	2	0.5	0.05	27.3	130	350	10	12
BZX84C43	Z80	42.1	43	43.9	2	0.5	0.05	30.1	150	375	10	12
BZX84C47	Z81	46.1	47	47.9	2	0.5	0.05	32.9	170	375	10	12
BZX84C51	Z82	50	51	52	2	0.5	0.05	35.7	180	400	10	12
BZX84C56	Z83	54.9	56	57.1	2	0.5	0.05	39.2	200	425	9	11
BZX84C62	Z84	60.8	62	63.2	2	0.5	0.05	43.4	215	450	9	12
BZX84C68	Z85	66.6	68	69.4	2	0.5	0.05	47.6	240	475	10	12
BZX84C75	Z86	73.5	75	76.5	2	0.5	0.05	52.5	255	500	10	12

Figure A.7: **TableDet inference result on sample image from Lytica test set**

A.2 Table Image Classification

The TableDet model variants (Model A and Model B) are trained for the task of table detection. Utilizing the enhanced table detection performances of these models we can also address the table image classification task as described in Section 5.1. Thus, the models do not require any additional training for this task. However, to evaluate the image classification performances with image classification models such as ResNet-50 [44] we have trained the image classifiers on the same training and test splits as the TableDet models as shown in Table 5.1. In Section A.2.1 we describe the datasets in more detail and provide the final results of TableDet Model A in Section A.2.2

A.2.1 Datasets

We utilize four public and one closed dataset to develop the public and private datasets as shown in Table 4.1. The TableDet model variants are trained on the public and private datasets in order to fairly compare the table detection performance improvements with state-of-the-art solutions that utilized similar datasets. In the following sections we describe each section in detail.

ICDAR 2013

The International Conference on Document Analysis and Recognition (ICDAR) in 2013 held a competition for two tasks; table detection and table structure detection [68]. To solve both task the competition provided a publicly available dataset of PDF government documents from European Union and United State of America. A total of 238 pages were extracted from these documents and converted into images. It is worth noting that none of these 238 images were used to train either image classifiers for table image classification nor the TableDet model for table detection as shown in Table 5.1 and Table 4.1 respectively. There are 128 and 110 Table and Non-Table images respectively in this test set.

ICDAR 2017

The ICDAR 2017 Page Object Detection competition [79] provided a dataset of 2417 images for the task of page object detection which involves detecting tables, text figures and mathematical formulas on the images. Each image was a page from academic reports available on CiteSeer. The 1600 training images in this set consisted of 557 and 1043 Table

and Non-Table images respectively as shown in Table 5.1. The official test set for this competition consists of 817 images. This test set consists of 244 Table and 573 Non-Table images. It is worth noting that the number of Non-Table images is close to double the number of Table images in both the training and test sets.

ICDAR 2019

The ICDAR 2019 Competition on Table Detection and Recognition [8] provided two subsets (modern and archival document pages) of images for the task of table detection. In this study we utilize the modern document subset of this dataset which comprises of 600 training and 240 test images. These images were extracted from academic and product reports and are a mix of English and Chinese document pages. Since this dataset only provided Table images we utilized only the training set for our analysis as shown in Table 5.1.

Marmot

The Peking University Computer Science Department released a 2000 image document full-page dataset for the task of table detection and has been utilized in multiple works such as Mark et al. [12], Jiang et al. [13] and Paliwal et al. [15]. The complete dataset comprises of 1000 English academic papers with two-column layout and 1000 Chinese pages from e-books with one-column layout. We maintained the same train and test splits as our analysis in table detection task. Thus, we utilized the entire Marmot dataset to train the TableDet model for table detection task and the image classifiers for table image classification task. This set comprises of 1006 Table and 994 Non-Table images.

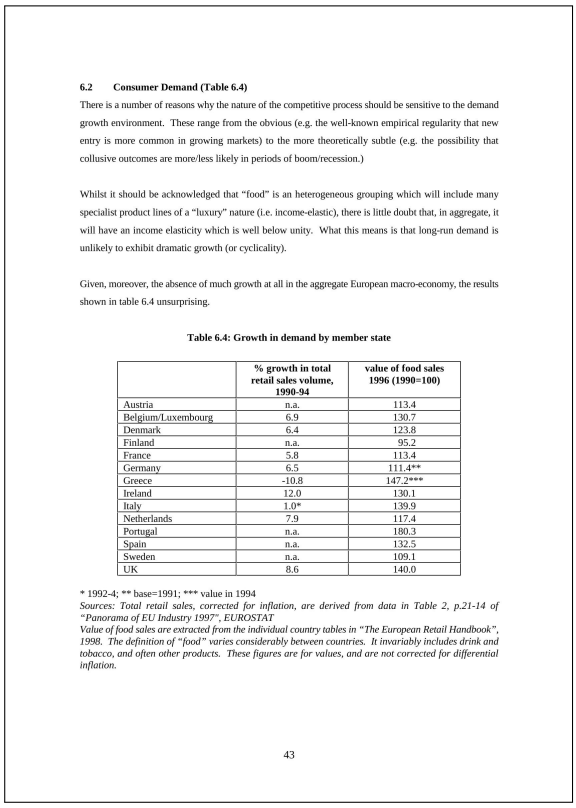
Lytica

Our industry partner, Lytic Inc. specializes in supply chain optimization and data analysis. They provided a commercial dataset consisting of multiple datasheet PDF documents with multiple types of tables. For instance, the some tables consisted of multiple row and columns with merged cells to display the tabular information in a hierarchical format. The dataset also consist of tables which have smaller tables within them. We converted 1240 pages into images and utilized it in our table detection analysis as shown in Table 4.1. For the task of table image classification we maintained the same train and test splits as our table detection analysis as shown in Table 5.1 and Table 4.1 respectively.

We trained the image classifiers on the training set of the public dataset and 1000 images from the Lytica dataset. These 1000 images were all Table images. However, the test set consist of 114 Non-Table images along with 126 Table images. We refer to this dataset as the private dataset as shown in Table 5.1. It is worth noting that unlike the public dataset, the Lytica dataset comprises of more Table images than Non-Table images. Furthermore, we utilized the Lytica dataset in a separate dataset which we refer to as the private dataset as shown in Table 5.1. Our analysis on the public and private datasets for table detection and table image classification can thus be fairly compared with state-of-the-art solutions that have utilized similar datasets in their develop their solutions.

A.2.2 Results

The TableDet Model A classifies the image as Table or Non-Table based on the three conditions in the classification head. It does not require any additional training to perform this task. In Fig A.8 we have displayed the inference results of True Postive detection by the mode. In Fig. A.9 we have displayed the False Positive detection and in Fig. A.10 we have displayed the True Negative detections.



6.2 Consumer Demand (Table 6.4)

There is a number of reasons why the nature of the competitive process should be sensitive to the demand growth environment. These range from the obvious (e.g. the well-known empirical regularity that new entry is more common in growing markets) to the more theoretically subtle (e.g. the possibility that collusive outcomes are more/less likely in periods of boom/recession.)

Whilst it should be acknowledged that "food" is an heterogeneous grouping which will include many specialist product lines of a "luxury" nature (i.e. income-elastic), there is little doubt that, in aggregate, it will have an income elasticity which is well below unity. What this means is that long-run demand is unlikely to exhibit dramatic growth (or cyclicality).

Given, moreover, the absence of much growth at all in the aggregate European macro-economy, the results shown in table 6.4 unsurprising.

Table 6.4: Growth in demand by member state

	% growth in total retail sales volume, 1990-94	value of food sales 1996 (1990=100)
Austria	n.a.	113.4
Belgium/Luxembourg	6.9	130.7
Denmark	6.4	123.8
Finland	n.a.	95.2
France	5.8	113.4
Germany	6.5	111.4**
Greece	-10.8	147.2***
Ireland	12.0	130.1
Italy	1.0*	139.9
Netherlands	7.9	117.4
Portugal	n.a.	180.3
Spain	n.a.	132.5
Sweden	n.a.	109.1
UK	8.6	140.0

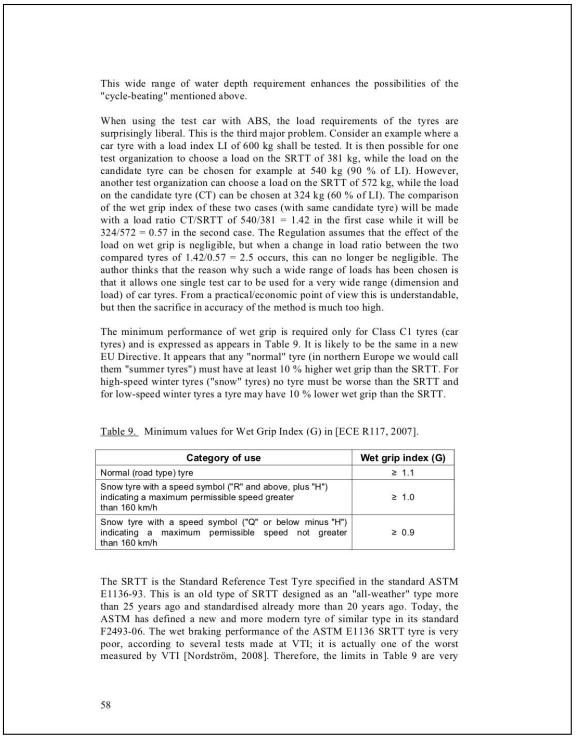
* 1992-4; ** base=1991; *** value in 1994

Sources: Total retail sales, corrected for inflation, are derived from data in Table 2, p.21-14 of "Panorama of EU Industry 1997", EUROSTAT

Value of food sales are extracted from the individual country tables in "The European Retail Handbook", 1998. The definition of "food" varies considerably between countries. It invariably includes drink and tobacco, and often other products. These figures are for values, and are not corrected for differential inflation.

43

(c)



This wide range of water depth requirement enhances the possibilities of the "cycle-beating" mentioned above.

When using the test car with ABS, the load requirements of the tyres are surprisingly liberal. This is the third major problem. Consider an example where a car tyre with a load index LI of 600 kg shall be tested. It is then possible for one test organization to choose a load on the SRTT of 381 kg, while the load on the candidate tyre can be chosen for example at 540 kg (90 % of LI). However, another test organization can choose a load on the SRTT of 572 kg, while the load on the candidate tyre (CT) can be chosen at 324 kg (60 % of LI). The comparison of the wet grip index of these two cases (with same candidate tyre) will be made with a load ratio CT/SRTT of 540/381 = 1.42 in the first case while it will be 324/572 = 0.57 in the second case. The Regulation assumes that the effect of the load on wet grip is negligible, but when a change in load ratio between the two compared tyres of 1.42/0.57 = 2.5 occurs, this can no longer be negligible. The author thinks that the reason why such a wide range of loads has been chosen is that it allows one single test car to be used for a very wide range (dimension and load) of car tyres. From a practical/economic point of view this is understandable, but then the sacrifice in accuracy of the method is much too high.

The minimum performance of wet grip is required only for Class C1 tyres (car tyres) and is expressed as appears in Table 9. It is likely to be the same in a new EU Directive. It appears that any "normal" tyre (in northern Europe we would call them "summer tyres") must have at least 10 % higher wet grip than the SRTT. For high-speed winter tyres ("snow" tyres) no tyre must be worse than the SRTT and for low-speed winter tyres a tyre may have 10 % lower wet grip than the SRTT.

Table 9. Minimum values for Wet Grip Index (G) in [ECE R117, 2007].

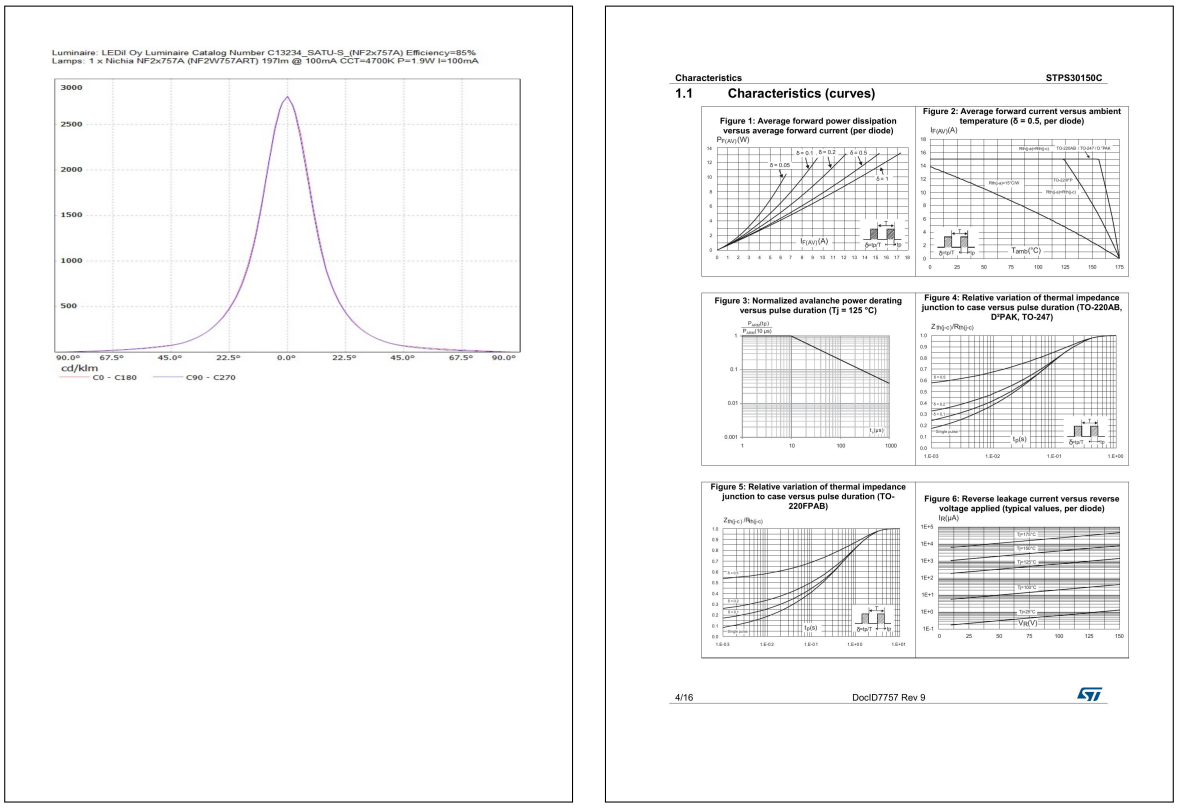
Category of use	Wet grip index (G)
Normal (road type) tyre	≥ 1.1
Snow tyre with a speed symbol ("R" and above, plus "H") indicating a maximum permissible speed greater than 160 km/h	≥ 1.0
Snow tyre with a speed symbol ("Q" or below minus "H") indicating a maximum permissible speed not greater than 160 km/h	≥ 0.9

The SRTT is the Standard Reference Test Tyre specified in the standard ASTM E1136-93. This is an old type of SRTT designed as an "all-weather" type more than 25 years ago and standardised already more than 20 years ago. Today, the ASTM has defined a new and more modern tyre of similar type in its standard F2493-06. The wet braking performance of the ASTM E1136 SRTT tyre is very poor, according to several tests made at VTI; it is actually one of the worst measured by VTI [Nordström, 2008]. Therefore, the limits in Table 9 are very

58

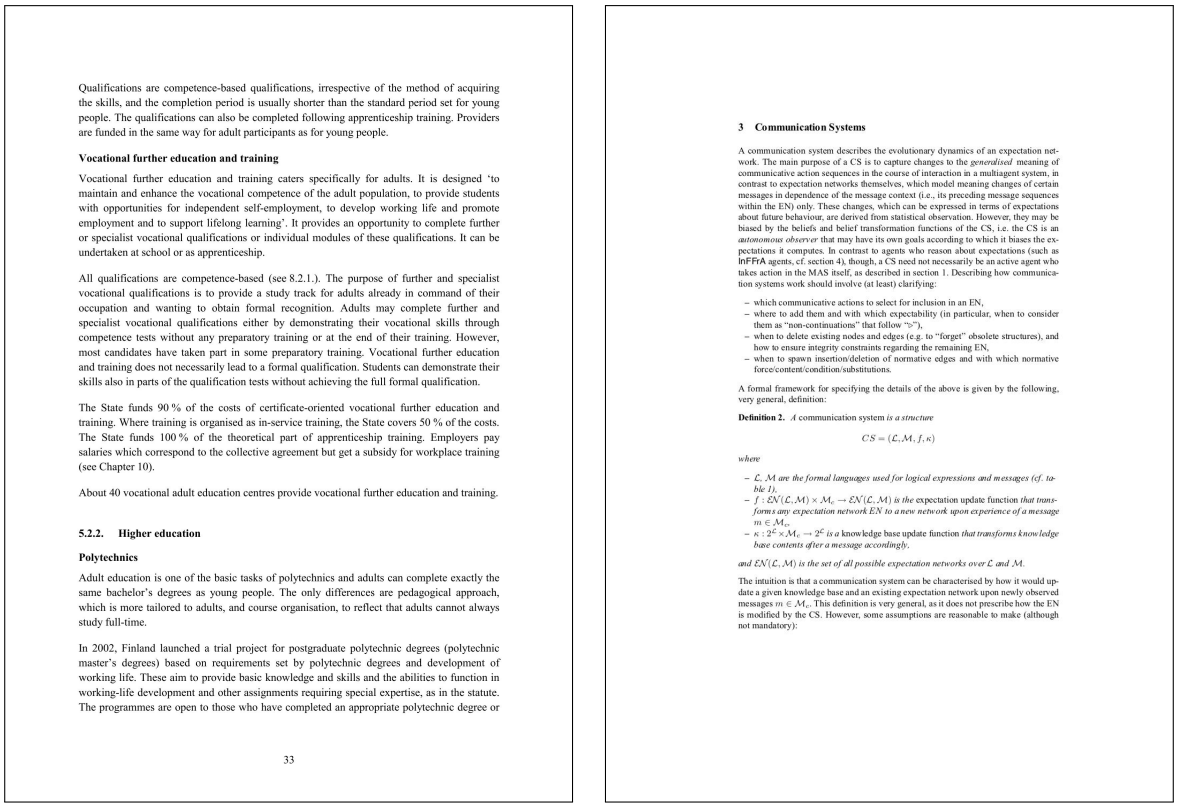
(d)

Figure A.8: **TableDet correctly classifies these Table images from the ICDAR 2013 test set (a) and ICDAR 2017 test set (b) as Table images (True Positive)**



(a) (b)

Figure A.9: **TableDet incorrectly classifies these Non-Table images from the Lytica test set as Table images (False Positive)**



Qualifications are competence-based qualifications, irrespective of the method of acquiring the skills, and the completion period is usually shorter than the standard period set for young people. The qualifications can also be completed following apprenticeship training. Providers are funded in the same way for adult participants as for young people.

Vocational further education and training

Vocational further education and training caters specifically for adults. It is designed 'to maintain and enhance the vocational competence of the adult population, to provide students with opportunities for independent self-employment, to develop working life and promote employment and to support lifelong learning'. It provides an opportunity to complete further or specialist vocational qualifications or individual modules of these qualifications. It can be undertaken at school or as apprenticeship.

All qualifications are competence-based (see 8.2.1.). The purpose of further and specialist vocational qualifications is to provide a study track for adults already in command of their occupation and wanting to obtain formal recognition. Adults may complete further and specialist vocational qualifications either by demonstrating their vocational skills through competence tests without any preparatory training or at the end of their training. However, most candidates have taken part in some preparatory training. Vocational further education and training does not necessarily lead to a formal qualification. Students can demonstrate their skills also in parts of the qualification tests without achieving the full formal qualification.

The State funds 90 % of the costs of certificate-oriented vocational further education and training. Where training is organised as in-service training, the State covers 50 % of the costs. The State funds 100 % of the theoretical part of apprenticeship training. Employers pay salaries which correspond to the collective agreement but get a subsidy for workplace training (see Chapter 10).

About 40 vocational adult education centres provide vocational further education and training.

5.2.2. Higher education

Polytechnics

Adult education is one of the basic tasks of polytechnics and adults can complete exactly the same bachelor's degrees as young people. The only differences are pedagogical approach, which is more tailored to adults, and course organisation, to reflect that adults cannot always study full-time.

In 2002, Finland launched a trial project for postgraduate polytechnic degrees (polytechnic master's degrees) based on requirements set by polytechnic degrees and development of working life. These aim to provide basic knowledge and skills and the abilities to function in working-life development and other assignments requiring special expertise, as in the statute. The programmes are open to those who have completed an appropriate polytechnic degree or

33

(c)

3 Communication Systems

A communication system describes the evolutionary dynamics of an expectation network. The main purpose of a CS is to capture changes to the *generalised* meaning of communicative action sequences in the course of interaction in a multiagent system, in contrast to expectation networks themselves, which model meaning changes of certain messages in dependence of the message context (i.e., its preceding message sequences within the EN) only. These changes, which can be expressed in terms of expectations about future behaviour, are derived from statistical observation. However, they may be biased by the beliefs and belief transformation functions of the CS, i.e. the CS is an *autonomous observer* that may have its own goals according to which it biases the expectations it computes. In contrast to agents who reason about expectations (such as INFFA agents, cf. section 4), though, a CS need not necessarily be an active agent who takes action in the MAS itself, as described in section 1. Describing how communication systems work should involve (at least) clarifying:

- which communicative actions to select for inclusion in an EN,
- where to add them and with which expectability (in particular, when to consider them as "non-continuations" that follow "o"),
- when to delete existing nodes and edges (e.g. to "forget" obsolete structures), and how to ensure integrity constraints regarding the remaining EN,
- when to spawn insertion/deletion of normative edges and with which normative force/content/condition/substitutions.

A formal framework for specifying the details of the above is given by the following, very general, definition:

Definition 2. *A* communication system *is a structure*

$$CS = (\mathcal{L}, \mathcal{M}, f, \kappa)$$

where

- $\mathcal{L}, \mathcal{M}$ *are the formal languages used for logical expressions and messages (cf. table 1),*
- $f : \mathcal{EN}(\mathcal{L}, \mathcal{M}) \times \mathcal{M}_c \to \mathcal{EN}(\mathcal{L}, \mathcal{M})$ *is the* expectation update function *that transforms any expectation network EN to a new network upon experience of a message* $m \in \mathcal{M}_c$*,*
- $\kappa : 2^{\mathcal{L}} \times \mathcal{M}_c \to 2^{\mathcal{L}}$ *is a* knowledge base update function *that transforms knowledge base contents after a message accordingly.*

and $\mathcal{EN}(\mathcal{L}, \mathcal{M})$ *is the set of all possible expectation networks over* $\mathcal{L}$ *and* $\mathcal{M}$*.*

The intuition is that a communication system can be characterised by how it would update a given knowledge base and an existing expectation network upon newly observed messages $m \in \mathcal{M}_c$. This definition is very general, as it does not prescribe how the EN is modified by the CS. However, some assumptions are reasonable to make (although not mandatory):

(d)

Figure A.10: **TableDet correctly classifies these Non-Table images from the ICDAR 2013 test set (a) and ICDAR 2017 test set (b) as Non-Table images (True Negative)**

A.3 Table Structure Detection Results

We propose TableStrDet, a deep learning solution to address the task of table structure detection in table images of datasheet (document) images. The TableStrDet model detects regular rows and irregular rows along with regular columns and irregular columns in order to capture the hierarchical layout of information in a table. Furthermore, as an additional step we utilize the intersections of these detected objects to generate accurate cell locations thus facilitating the next step in the text extraction pipeline; tabular text extraction. In Section A.3.1 we describe the datasets utilized in this study in more detail. Furthermore, we display the inference results of TableStrDet in Section A.3.2

A.3.1 Datasets

We utilize two datasets in this work to train the TableStrDet model to accurately detect and classify the rows and columns of each table image. Both datasets are available publicly and we ensured to utilize the same train and test splits in order for us to fairly compare out table structure detection performances with state-of-the-art systems.

ICDAR 2013

The ICDAR 2013 dataset [68] contains 67 American and European Union documents. There are a total of 238 pages combining all documents of the dataset which are converted to images. In this work we only use the table regions of the full page images. Thus we utilize only 156 tables provided by this dataset to train and test the table structure detection methods. We maintain the same training and test split as Siddiqui et al. [56] to train and evaluate our proposed method. As shown in Table 6.1, 125 tables of ICDAR 2013 dataset are used for training and 31 tables for testing. Each image in the dataset has annotations for regular rows and regular columns. If a merged cell is present in the table the annotations for irregular rows and irregular columns are also provided.

TabStructDB

The TabStructDB was developed by Siddiqui et al. [56] who utilized the ICDAR 2017 POD competition dataset [79] to extract a total of 1081 tables. We have maintained annotations for regular rows and regular columns for each image of this dataset. Furthermore, irregular rows and irregular columns annotations are provided for images where a merged

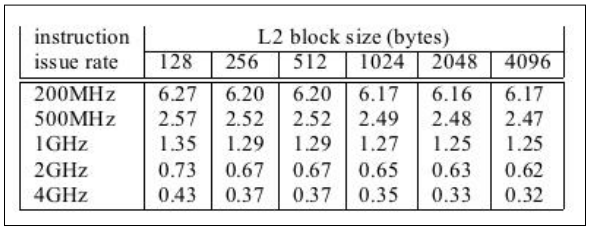
cell is present. To fairly compare the table structure detection performance of our proposed method with the state-of-the-art method we maintain the same training and test split as Siddiqui et al. [56]. Thus, 731 tables and 350 tables from TabStructDB are used for training and testing, respectively as shown in Table 6.1.

A.3.2 Results

We propose TableStrDet, a deep learning based approach to detect and classify rows and columns in table images. Fig. A.11 (a), Fig. A.12 (a) and Fig. A.13 (a) are the input to the TableStrDet model. As shown in Fig. A.11 (b) and Fig. A.11 (c) TableStrDet can accurately detect and classify rows and columns on table images where line separators exist between columns and not between rows. We denote the irregular columns and regular columns in Fig. A.11 (b) in blue colour dotted line box and red colour line box respectively. The detected irregular rows and regular rows in Fig. A.11 (c) are shown in blue colour dotted line box and red colour line box respectively.

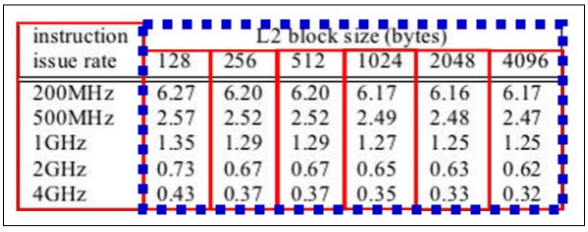
Furthermore in Fig. A.12 (b) and Fig. A.12 (c) we display the detection results on a table image with some line separators in the row header while the rest of the table does not have any line separators. We display the detected irregular columns and regular columns in Fig. A.12 (b) with a blue colour dotted line box and red colour line box respectively. The detected irregular rows and regular rows in Fig. A.12 (c) are shown in blue colour dotted line box and red colour line box respectively.

Fig. A.13 (a) is an input image with line separators exist between all cells of the image. Result of post-processing step where the intersections of detected irregular columns, regular columns, irregular rows and regular rows are used to generate cell locations are shown in Fig. A.11 (d), Fig. A.12 (d) and Fig. A.13 (d).



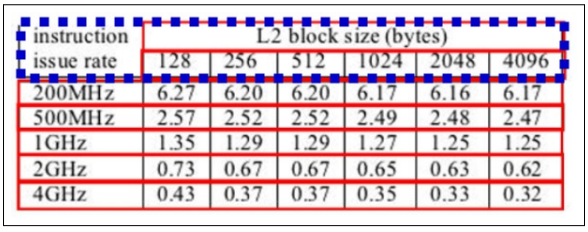
instruction issue rate	L2 block size (bytes)					
	128	256	512	1024	2048	4096
200MHz	6.27	6.20	6.20	6.17	6.16	6.17
500MHz	2.57	2.52	2.52	2.49	2.48	2.47
1GHz	1.35	1.29	1.29	1.27	1.25	1.25
2GHz	0.73	0.67	0.67	0.65	0.63	0.62
4GHz	0.43	0.37	0.37	0.35	0.33	0.32

(a)



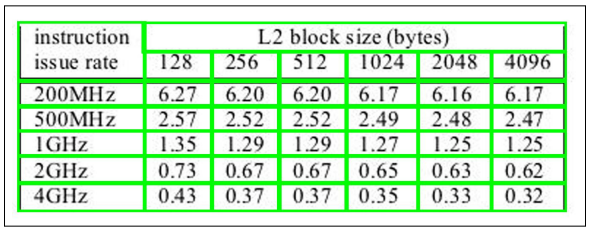
instruction issue rate	L2 block size (bytes)					
	128	256	512	1024	2048	4096
200MHz	6.27	6.20	6.20	6.17	6.16	6.17
500MHz	2.57	2.52	2.52	2.49	2.48	2.47
1GHz	1.35	1.29	1.29	1.27	1.25	1.25
2GHz	0.73	0.67	0.67	0.65	0.63	0.62
4GHz	0.43	0.37	0.37	0.35	0.33	0.32

(b)



instruction issue rate	L2 block size (bytes)					
	128	256	512	1024	2048	4096
200MHz	6.27	6.20	6.20	6.17	6.16	6.17
500MHz	2.57	2.52	2.52	2.49	2.48	2.47
1GHz	1.35	1.29	1.29	1.27	1.25	1.25
2GHz	0.73	0.67	0.67	0.65	0.63	0.62
4GHz	0.43	0.37	0.37	0.35	0.33	0.32

(c)



instruction issue rate	L2 block size (bytes)					
	128	256	512	1024	2048	4096
200MHz	6.27	6.20	6.20	6.17	6.16	6.17
500MHz	2.57	2.52	2.52	2.49	2.48	2.47
1GHz	1.35	1.29	1.29	1.27	1.25	1.25
2GHz	0.73	0.67	0.67	0.65	0.63	0.62
4GHz	0.43	0.37	0.37	0.35	0.33	0.32

(d)

Figure A.11: **TableStrDet inference results on sample images from TabStructDB test set**

	NRMSE		MND	
Domain	G&T-Bayesian	Min2	G&T-Bayesian	Min2
Cake baking	0.38	**0.17**	1.79	**1.13**
Congressional terms	0.20	**0.16**	1.64	**1.10**
Movie grosses	0.20	**0.17**	2.12	**1.83**
Waiting times	0.08	**0.05**	0.52	**0.25**
Poem lengths	0.15	**0.13**	1.20	**0.94**
Movie run times	1.06	1.21	1.73	**1.69**
Pharaoh reigns	0.33	0.45	0.47	0.74
Lifespans	0.26	1.26 (**0.20**)	2.39	4.71 (**1.26**)

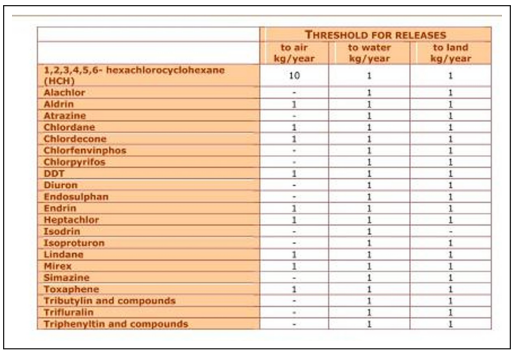
(a)

(b)

(c)

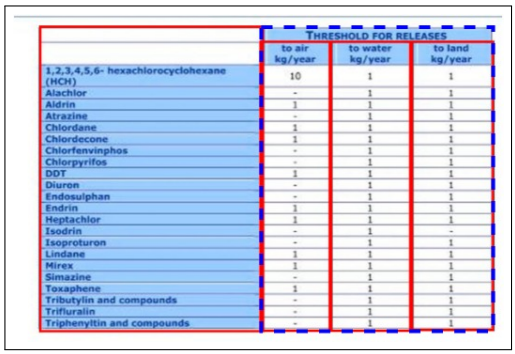
(d)

Figure A.12: **TableStrDet inference results on sample images from TabStructDB test set**



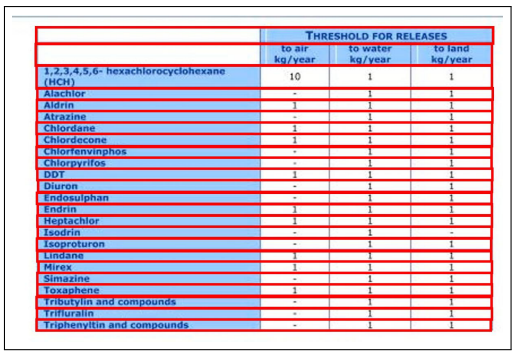
	THRESHOLD FOR RELEASES		
	to air kg/year	to water kg/year	to land kg/year
1,2,3,4,5,6- hexachlorocyclohexane (HCH)	10	1	1
Alachlor	-	1	1
Aldrin	1	1	1
Atrazine	-	1	1
Chlordane	1	1	1
Chlordecone	1	1	1
Chlorfenvinphos	-	1	1
Chlorpyrifos	-	1	1
DDT	1	1	1
Diuron	-	1	1
Endosulphan	-	1	1
Endrin	1	1	1
Heptachlor	1	1	1
Isodrin	-	1	-
Isoproturon	-	1	1
Lindane	1	1	1
Mirex	1	1	1
Simazine	-	1	1
Toxaphene	1	1	1
Tributylin and compounds	-	1	1
Trifluralin	-	1	1
Triphenyltin and compounds	-	1	1

(a)



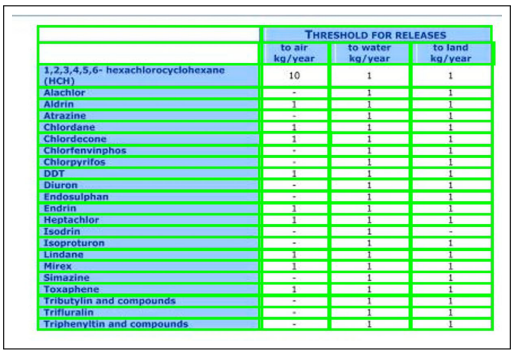
	THRESHOLD FOR RELEASES		
	to air kg/year	to water kg/year	to land kg/year
1,2,3,4,5,6- hexachlorocyclohexane (HCH)	10	1	1
Alachlor	-	1	1
Aldrin	1	1	1
Atrazine	-	1	1
Chlordane	1	1	1
Chlordecone	1	1	1
Chlorfenvinphos	-	1	1
Chlorpyrifos	-	1	1
DDT	1	1	1
Diuron	-	1	1
Endosulphan	-	1	1
Endrin	1	1	1
Heptachlor	1	1	1
Isodrin	-	1	-
Isoproturon	-	1	1
Lindane	1	1	1
Mirex	1	1	1
Simazine	-	1	1
Toxaphene	1	1	1
Tributylin and compounds	-	1	1
Trifluralin	-	1	1
Triphenyltin and compounds	-	1	1

(b)



	THRESHOLD FOR RELEASES		
	to air kg/year	to water kg/year	to land kg/year
1,2,3,4,5,6- hexachlorocyclohexane (HCH)	10	1	1
Alachlor	-	1	1
Aldrin	1	1	1
Atrazine	-	1	1
Chlordane	1	1	1
Chlordecone	1	1	1
Chlorfenvinphos	-	1	1
Chlorpyrifos	-	1	1
DDT	1	1	1
Diuron	-	1	1
Endosulphan	-	1	1
Endrin	1	1	1
Heptachlor	1	1	1
Isodrin	-	1	-
Isoproturon	-	1	1
Lindane	1	1	1
Mirex	1	1	1
Simazine	-	1	1
Toxaphene	1	1	1
Tributylin and compounds	-	1	1
Trifluralin	-	1	1
Triphenyltin and compounds	-	1	1

(c)



	THRESHOLD FOR RELEASES		
	to air kg/year	to water kg/year	to land kg/year
1,2,3,4,5,6- hexachlorocyclohexane (HCH)	10	1	1
Alachlor	-	1	1
Aldrin	1	1	1
Atrazine	-	1	1
Chlordane	1	1	1
Chlordecone	1	1	1
Chlorfenvinphos	-	1	1
Chlorpyrifos	-	1	1
DDT	1	1	1
Diuron	-	1	1
Endosulphan	-	1	1
Endrin	1	1	1
Heptachlor	1	1	1
Isodrin	-	1	-
Isoproturon	-	1	1
Lindane	1	1	1
Mirex	1	1	1
Simazine	-	1	1
Toxaphene	1	1	1
Tributylin and compounds	-	1	1
Trifluralin	-	1	1
Triphenyltin and compounds	-	1	1

(d)

Figure A.13: **TableStrDet inference results on sample images from ICDAR 2013 test set**