Master's Thesis

Extraction of Solar Cell Data from PDF Datasheets

Muhammad Moeez Malik

Examiner: Prof. Dr. Hannah Bast
Advisers: Dr.-Ing. Christian Reichel
Dr. Patrick Brosi

University of Freiburg
Faculty of Engineering
Department of Computer Science
Chair for Algorithms and Data Structures

Fraunhofer ISE
Fraunhofer-Institut für Solare Energiesysteme
Freiburg im Breisgau

April 11th, 2023

Writing Period

18. 01. 2023 – 11. 04. 2023

Examiner

Prof. Dr. Hannah Bast

Advisers

Dr.-Ing. Christian Reichel, Dr. Patrick Brosi

Declaration

I hereby declare that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

_____ _____

Place, Date Signature

Abstract

Tables in data sheets contain the most critical information. Due to variability in design of data sheets, computers cannot readily process and extract data from them. In this thesis, an end-to-end pipeline is presented that aims to extract solar cell data from tables in PDF data sheets. The approach is divided into three major parts. Firstly, the table containing areas of the data sheets are detected using deep learning based object detectors. Secondly, the raw values contained in the tables are extracted using off-the-shelf table extraction utilities. Finally, the raw values are structured and validated using a combination of machine learning and rule-based methods. The complete pipeline achieved 99.86% precision and 94.9% recall performance at extracting electrical and thermal characteristics from solar cell data sheets.

Contents

List of Figures

List of Tables

1 Introduction

We are living in the information age. In recent years, the interest in collecting, organising and analysing data to make processes more efficient and to make better informed decisions has been increasing. Hence, there is a strong motivation to harvest data from existing sources of information. This process is called data mining. We are surrounded by resources of data that are just waiting to be mined. The problem, however, is that over the years, all of these resources have been designed to be processed and consumed by humans which makes it hard for computers to understand them.

One particular example of this are documents. Even in digital formats (one of the most popular one being PDF), the documents are designed to be consumed by humans. Due to the lack of standardisation in information design, it becomes very hard for a computer to automatically process them and extract relevant information. If a process can be devised that allows the computers to automatically process these documents, we can quickly and efficiently amass a great deal of useful data. An example of this process is illustrated in Figure 1.

1.1 Motivation

Due to the popularity of the modern digital file format PDF, much of the information about different products in the market is distributed through them. One prime

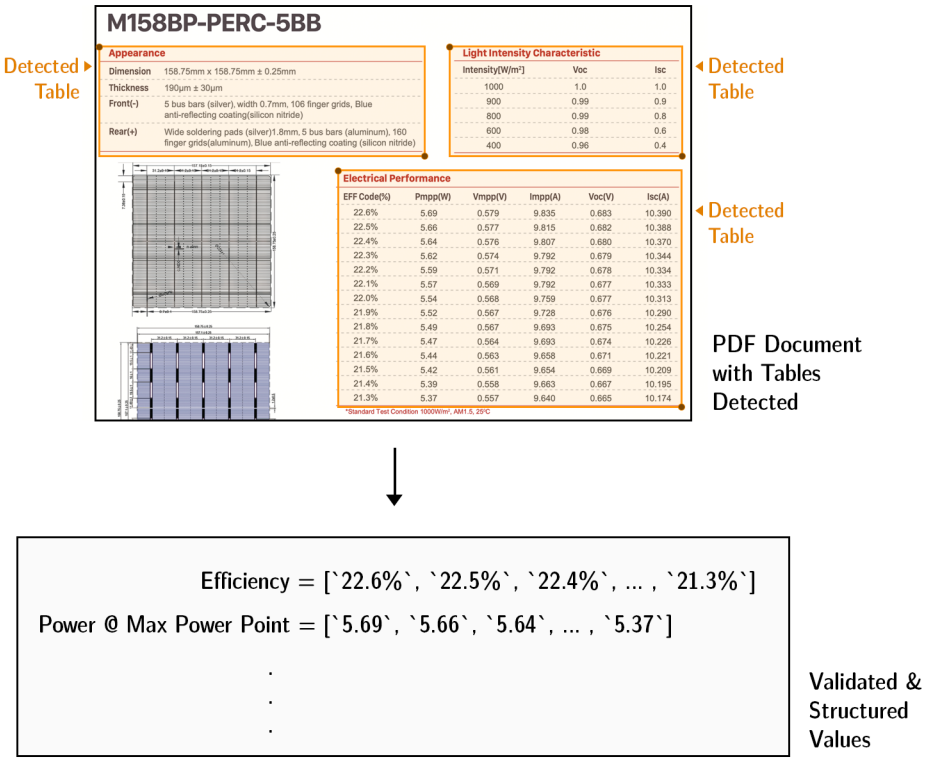


Figure 1: Extracting information from a data sheet The figure illustrates the process of extracting electrical characteristics from a solar cell data sheet. The detected tables are indicated by orange bounding boxes. Here, the column 'EFF Code(%)' is mapped to 'Efficiency' and the column 'Pmpp(W)' is mapped to 'Power @ Max Power Point'. Figure 18 through Figure 21 show this mapping procedure in more detail.

example for this are the data sheets. The particular type of data sheets that are of importance for the task at hand are the data sheets for solar cells. The goal is to extract the solar cell specifications presented in tabular format from these data sheets. Of course, this information can be extracted manually and saved to a database. That is fine for a couple of documents but needless to say, this approach is not feasible for a large quantity of documents.

The information contained in the data sheets, such as electrical characteristics of the solar cell which are always mentioned in a table format, can be of critical importance especially for research purposes. By developing a database of searchable and indexed

information, a market wide survey can be generated that can identify market trends and eventually can uncover the areas that are under-researched. Thus, the motivation for undertaking this thesis was to develop an end-to-end pipeline that is able to process solar cell data sheets and extract specifications mentioned in tables regardless of the manufacturer and styling of the data sheet.

1.2 Problem Statement

There are numerous advantages of using PDF files. Their ability to render the exact same way regardless of the operating system and hardware makes them appealing for the type of documents where accurate dissemination of information is critical. For this reason, PDF has become the default format through which manufacturers dispense information about their products in the digital era. Having said that, they are primarily designed for humans to read which is why they can lack a structure that makes them very hard to be understood by computers.

Using the PDF format, the manufacturers can ensure information security and customise the documents according to their branding. Each manufacturer, internally, has its own philosophy about how to present the information and that is reflected in their documents. Primarily, for the case of data sheets at least, the information is presented in the form of tables but each manufacturer has its own way of designing and styling the tables. This results in a lot of variability between styling of the documents which is fine for humans to understand but hard for machines.

Due to the variability in the documents, it is hard to devise a rule-based algorithm that is able to detect and extract information from the tables in these documents. There are open-source rule-based projects for Python available such as Camelot and Tabula that aim to solve the problem of table detection and extracting information from them. However, in an earlier analysis [1], they failed at detecting tables in solar data sheets. This is because these projects are not made for extracting tables from

solar data sheets, where there are 5 to 6 tables per page on average, but rather simple documents which usually contain only a single table per page.

In addition to the problem of locating the tables, manufacturers can use a range of terms to denote the column or row names in the tables which makes it hard to locate the values of interest. The title indicating the type of table may be omitted as well and that can create problems in identifying the type of a table. During the process of locating and understanding tables, there could also be errors and occlusions in the extraction of values. Thus, an additional step is needed that can process the raw values extracted by table detection and table recognition.

Keeping these problems in view, an approach needs to be devised that can first and foremost detect (locate) the tables in the documents well. After the tables have been detected, the rows and columns need to be correctly identified and all values from the table need to be extracted. A final step is also needed that can make up for the errors in the previous steps and validate extracted values.

1.3 Overview

This thesis is divided into multiple chapters in order to present the work undertaken in an organised fashion. A brief introduction to these chapters is as follows.

- In Chapter 2, a literature survey of works relevant and related to the task at hand are mentioned.

- In Chapter 3, a brief introduction to all the technologies used for the approach are outlined.

- In Chapter 4, the methodology taken to solve the individual problems is described.

- In Chapter 5, the results from the experiment run in order to test the effectiveness of the methodologies are presented.

- In Chapter 6, the final conclusions about the approach taken to solve the problems at hand are drawn. Future work and possible improvement pathways are also mentioned.

2 Related Work

The problem of detecting and understanding tables in documents is well known and previous work has been done to tackle it. The most relevant of these works to the required task will be presented in this chapter. The first three sections of the chapter will present the related works for the three main problems being solved and then finally the contributions achieved by this thesis will be highlighted.

2.1 Table Detection

The first step in extracting information from the tables is to detect regions in the documents that contain the tables. Thus the problem of detecting (locating) the tables in the documents is well known and there has been some research that aims to solve it.

One of the earlier work in the domain of rule-based methods was presented by Shamilian et al. [2]. They presented an approach to detect tables based on the specified layouts. They also presented a GUI that allowed the user to re-program the target layout. Klein et al. [3] presented three different approaches for spotting tables in industrial documents. Hassan et al. [4] presented a rule-based approach for detecting tables with and without ruling lines forming the tables. These approaches, however, depend on pre-definition of the layout of the tables and are hard to generalise when the table layouts change.

Wang et al. [5] presented an optimisation based strategy for segmenting the document into different regions that can also indicate the presence of tables. Recently, there has been interest in using Deep Learning for this task as well. Hao et al. [6] used Convolutional Neural Networks to classify the proposed regions as either a table or not. In the same footsteps, Gilani et al. [7] applied FasterRCNN as object detectors for table detection. FasterRCNN was also used by Schreiber et al. [8] not only for table detection but also table recognition.

2.2 Table Recognition

Table Recognition (sometimes also referred to as Table Understanding) is the step that identifies the rows and columns of the table. In some approaches, Table Recognition is combined with Table Detection such as in the work presented by Hassan et al. [4], Wang et al. [5] and Schreiber et al. [8] with the last approach being based on Deep Learning. The master thesis work done by Nurminen [9] presented another rule-based method for table recognition that has been used as the basis of one popular table processing utilities for Python named Camelot. A clustering based technique was presented by Zucker et al [10] that identifies and groups text boxes in tables into correct rows and columns using horizontal and vertical clustering techniques.

Raja et al. [11] presented a deep learning based object detector to recognise the structure of tables. They targeted the problem of correctly identifying empty cells to improve recognition of multi-row or multi-column cells. Qasim et al. [12] presented an approach that formulated the problem of table recognition to be solved by Graph Neural Networks. Both traditional and deep learning based works have been discussed here for the Table Recognition problem. Deep Learning based methods can perform well on unseen data but they have the caveat of needing a great deal of training data to achieve good performance which is not always available especially for specific applications.

2.3 Text Classification

Text classification is the process of categorising text into one of the predefined classes. This technique will be utilised to detect the type of the table based on its textual content. Previous works tackling the problem of text classification presented here can be classified into two categories: Classical machine learning and deep learning based approaches. While the deep learning pipelines can be end-to-end, the classical machine learning approaches are further sub-divided into two steps known as Text Representation and Classification.

Text representation is the process of converting the text in a form that is more easily processed by computers. There are a number of proposed approaches to do that. Bag of Words [13], N-grams [14] and TF-IDF scores [15] are simple text representation techniques for this purpose that mainly take into account the frequency of occurrence of words. Word2Vec [16] and GloVe [17] are more sophisticated as the text representations generated by these approaches are closer in distance for similar words. Hence, they can embed semantic similarity into the text representation as well.

In the second step of classification, classifiers that are commonly used are Naive Bayes [18] and K Nearest Neighbours [19]. Joachims et al. [20] also presented a way to use Support Vector Machines for the task of text classification.

Recent advancements in deep learning has also resulted in some interesting applications in the area of text classification. Convolutional Neural Networks are usually used for image-based data but the work done by Kim et al. [21] modified them to perform sentence classification as well. Recurrent Neural Networks are a type of neural networks that can take in a sequence of inputs which makes them ideal for text-based applications. They have also been utilised for the purpose of text classification [22]. Due to the large number of trainable parameters, for deep learning based approaches

to perform well, they need large amounts of training data which might not always be available for task specific applications.

2.4 Contributions

During the course of this thesis, the following contributions were achieved for the required task of extracting information from PDF data sheets automatically.

- Created a labelled dataset for table detection in solar cell and module data sheets.
- Trained and evaluated Deep Learning based Object Detectors for detecting tables in solar data sheets.
- Compared off-the-shelf table recognition utilities on the task specific dataset.
- Trained, evaluated and compared text classification approaches for classifying the tables from their raw content.
- Created processes to locate and validate the required values from the raw values extracted from the tables.
- Created and evaluated an end-to-end pipeline using the individual components specified above that can take in PDF documents and extract the values of interest from the tables within.

3 Background

In order to follow the approach towards solving the problem of table data extraction, it is necessary to understand the underlying technologies that will form the methodology. This chapter presents, in brief detail, the background of these technologies, starting from the description of the PDF format and following up with machine learning techniques that were used.

3.1 Portable Document Format (PDF)

3.1.1 What are PDF documents?

If you have ever hand-crafted a digital document with a custom font and layout only to have it appear differently on another computer, then you are familiar with the struggle. Ordinary rich text documents only contain the crucial information regarding the text to keep the size small, thus the reader software on the receiving end decides how to display the document. The PDF project, started by Adobe in 1991 [23], aimed to solve this problem. It defined a set of standards that make it possible for a document to appear exactly the same on all computers regardless of hardware and the operating system. This is the reason that much of the information provided by manufacturers is in the form of PDF documents these days. Using a PDF they can ensure that end-user get the document that is rendered accurately every time.

It is important to understand the composition of a PDF document since solar manufacturers also distribute the information using PDF documents and because the goal is to extract relevant information from them. A PDF document is made up of only a few entities [24]. These entities are as follows:

- **Text:** This entity type not only contains the raw text but also other information that is needed to show it in the document such as the font and the colour of the text.

- **Vector graphics:** These are graphics that are stored in the form of mathematical epxressions and can be drawn at every scale of the document.

- **Raster images:** These are just raw images that sometimes need to be added to the documents. Raster images cannot be scaled like vector graphics and will pixelate when zoomed in.

- **Multimedia objects:** PDF also supports multimedia object such as video.

Each of the entities that are mentioned above not only contain their own respective information but also their positional information in the document. This way a PDF reader can take in the positional information from all of these entities and display them at the correct spots in the documents. This way the document is rendered the same way on all devices.

All of the these entities can be considered to be on separate layers. As discussed in subsequent chapters, there will be a need to render these PDF documents not as separate layers but rather as flattened down images so that image based object detectors (discussed in later section) can process them. Since, PDF is now a well established standard, there are a number of open source projects available that can help in rendering these separate layers into a flattened image. One such project is PyMuPDF [25].

In day to day practice, however, there is a variety of PDF documents that might not contain all these aforementioned entities. If you have ever used a scanner to scan a physical document into a PDF file then you might realise that all of the text of that physical document will be in the form of an image (raster image). In this case the text layer of the PDF does not contain anything as we only have the snapshot of the text of the physical document. This is one caveat that needs to be take into consideration while processing the data sheets. The open source libraries that will be used later on can only work on the type of PDFs that contain this text layer. A large quantity of data sheets do contain this text layer since they are generated directly from a computer rather than scanned, hence it is worthwhile to develop a pipeline that can only process the text based PDFs. To make the processing of image-based documents possible, a process known as Optical Character Recognition (OCR) can be incorporated into the pipeline. However, that was not investigated as part of this thesis.

3.1.2 Extracing Information from PDFs

As discussed above, the PDF format is useful for manufacturers to distribute information. Since the information is organised in layers, it is very easy to extract raw textual information from the PDFs. This raw textual information is unstructured data which is hard for computers to understand. In addition, in case of data sheets, most of the information is presented in the form of tables. Reading tables comes naturally to humans but it is not straight forward for computers. The extraction of raw information from tables is a two-step process. These steps are introduced below.

Table Detection is the process of locating the table in a PDF document. The output of a table detection process are the x-y coordinates that represent the location of the table. It is easy for humans to spot the tables by analysing visual cues. It can also be done across data sheets from different manufacturers. Certain rules can be

devised for computers to look for these visual cues for a single manufacturer but it is very hard to do so for all manufacturers.

Table Recognition Once the location of the table is determined in the PDF, the next step is to determine the rows and cells of the tables and extract the values. This step is called Table Recognition. Again, this is easy for humans but due to variations in the table layout across different manufacturers (e.g. merged cells), it is not so easy for computers.

Therefore, there is a need to optimise both of these steps for increased success of extracting correct values from the tables.

3.2 A Brief Introduction to Machine Learning

This section will provide a brief introduction to Machine Learning. By first presenting simple rule-based methods and their shortcomings, a motivation for machine learning systems will be developed. Later, machine learning as an approach to Artificial Intelligence will be discussed.

3.2.1 Rule-Based Methods

Computers are very good at following explicit instructions. They are able to perform mathematical computations accurately and quickly while holding large amounts of data in the working memory. In this aspect they are better than humans. When trying to solve any problem using computers, the first attempt is to create a set of rules for it to follow to achieve the desired results. For very simple problems such as the algorithms running your washing machine, it makes sense. The desired output is produced most of the times if not always. Such simple methods are known as rule-based methods.

The problem occurs when something unknown happens. This is where computers are not very good. They are not creative, hence they cannot deal with incoming data that is out of the realm of the set of instructions defined by rule-based methods. Thus, rule-based methods face severe generalisation problems. In this case, it would be helpful if the computer can generate its own set of rules based on the specific problem rather than trying to cater for each and every type of input. This is the promise of learning-based algorithms which will be explored in the following sections.

3.2.2 Artificial Intelligence

Humans have been trying to create machines that can think on their own for centuries [26], a fantasy that has been expressed through countless work of science fiction books and movies. Even today, the true thinking machines (which fall under the paradigm of Artificial General Intelligence, intelligence that can rival that of humans) is still a figment of imagination. Having said that, due to the recent advancements in the technology and thanks to ever-increasing powers of our computers, there are a certain tasks on which computers can demonstrate human-like performance. All of this has been possible due to the advancements in the field of artificial intelligence. The advantages of creating machines that can think are obvious, they will be able to automate the tasks that are laborious for humans and for which rule-based methods are not sufficient.

As discussed above, computers are very good at following instructions but not so good at creativity and learning automatically from past experiences as humans do. The most difficult tasks for computers to perform are the ones that are very intuitive for humans. We call it intuition because we are still yet to fully understand how our brains work and why these tasks are very easy for us. For example, speech recognition, driving a car while paying attention to everything that is on the road and detecting tables in solar cell and module data sheets. Artificial Intelligence can help in this case

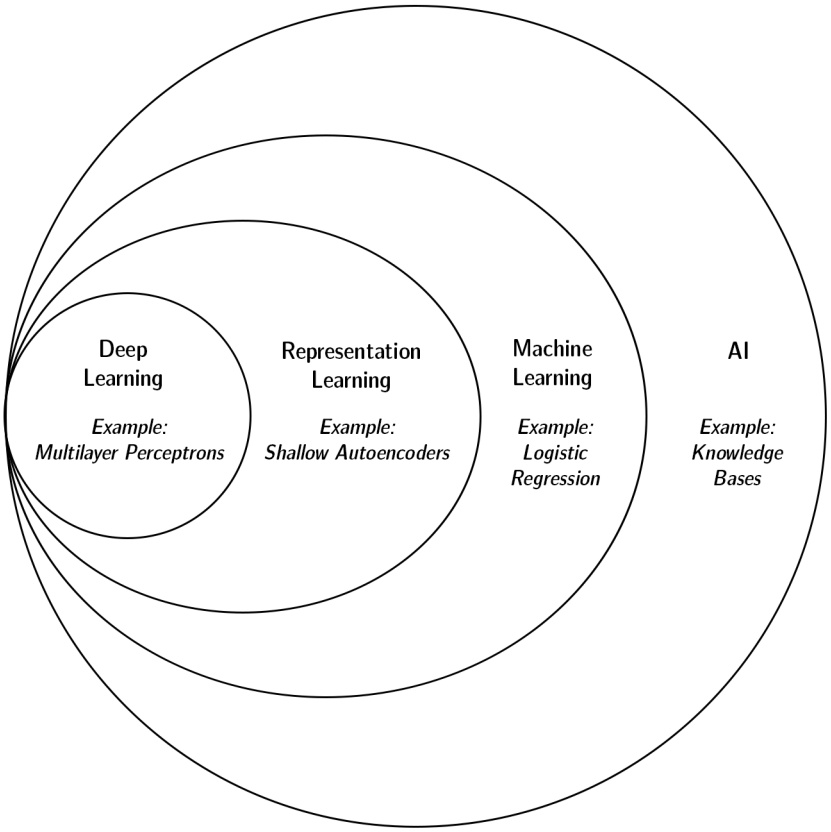


Figure 2: Venn Diagram of Artificial Intelligence (AI) Methods The figure that shows the Venn diagram of present day approaches to Artificial Intelligence. Figure is taken from the Deep Learning Book [26]

and Machine Learning methods is one of the approach towards artificial intelligence by imitating the way humans learn.

3.2.3 Machine Learning

One of the core principles of machine learning is to try and make the computer learn the way humans learn. We, as humans, can tap into past experiences and learn from them. The idea for machine learning is similar, we design the algorithms that are

able to learn from the past data and can make predictions on new unseen data. Most things in life obey a certain order e.g. the sun rises from the East and sets in the West. Hence studying from the past data and learning from it allows the computer to make intelligent decisions and predictions.

Machine Learning methods are ubiquitous in today's world. You have likely interacted with it multiple times today already. It is used by recommendation engines of all major social media platforms, by Netflix for recommending you new movies and by Youtube to recommend what new videos to watch next. The field of machine learning (specifically deep learning, which we will discuss later) has also revolutionised major areas of computer vision. Computer Vision as the name suggests deals in giving the computers the perception of vision. The advancements made in the field of Computer Vision will be heavily utilised for one of the required tasks as well i.e. Detecting Tables in the PDF documents.

Unsupervised Machine Learning

In the paradigm of Unsupervised Learning, the machine is asked to create its own representation of data. The data provided to the machine is not labelled, which essentially means that an expert e.g. a human has not told the machine any additional information about the data. Instead, the computer is asked to make sense of the data on its own. Unsupervised learning can be thought of as asking the machine to learn without any teacher. This results in some difficulties. As there is no enforcer, the results might be inaccurate. In addition, there is a lack of transparency on the decision making process. One of the major applications of Unsupervised Learning is in the area of clustering where the machine is asked to group together data points that are similar.

Supervised Machine Learning

In contrast to Unsupervised Learning is the paradigm of Supervised Learning. Here, the data that is inputted to the machine is labelled i.e. an expert such as a human has told the computer what should be the expected output for a certain input datapoint. In this way when the computer makes a mistake while performing the task, it can be told that the result is not right. This will result in some changes of internal parameters of the algorithm that will make it better over time. All of the machine learning techniques that will be used for the required tasks, as will observed later, belong to the paradigm of Supervised Learning.

3.2.4 Classical vs. Modern Machine Learning

In Section 3.2.1 we discussed the rule-based methods that are based on rigid rules in order to perform a certain task. We already know that these tasks do not generalise well on new data and this is where machine learning comes in. Before the era of modern algorithms, due to lack of computing resources, the machine learning methods were the ones where humans helped the algorithms in forming a representation of the data. These class of algorithms are now referred to as Classical or Traditional Machine Learning algorithms.

In classical methods, for any task that needed to be performed through Machine Learning, a domain expert will need to understand the data and hand craft the features that will then be used by the algorithms to perform the task. These hand crafted features are the so-called "hidden" representation of the data. These techniques are better than rule-based methods for making predictions on new data but these methods can be arduous since it needs extensive feature engineering to get the right results.

Representation Learning

What if, instead of a human creating the representation (features) of the data, the computer can itself form the required representation based on the task? That is precisely the goal of Representation Learning methods.

The clear advantage of such methods is that the dependence on a domain expert for achieving good results for the task is greatly reduced. Given the data, the computers might be able to find a hidden representation that results in better performance than what even a human feature engineers. However, this also comes with the downside, since the representation is learned automatically, humans cannot make any sense of it. This results in a lack of transparency of how the algorithms are coming to their decision.

One class of representation learning algorithms that have taken over the world recently and have revolutionised a number of fields already are Deep Learning methods. These will be discussed in detail later.

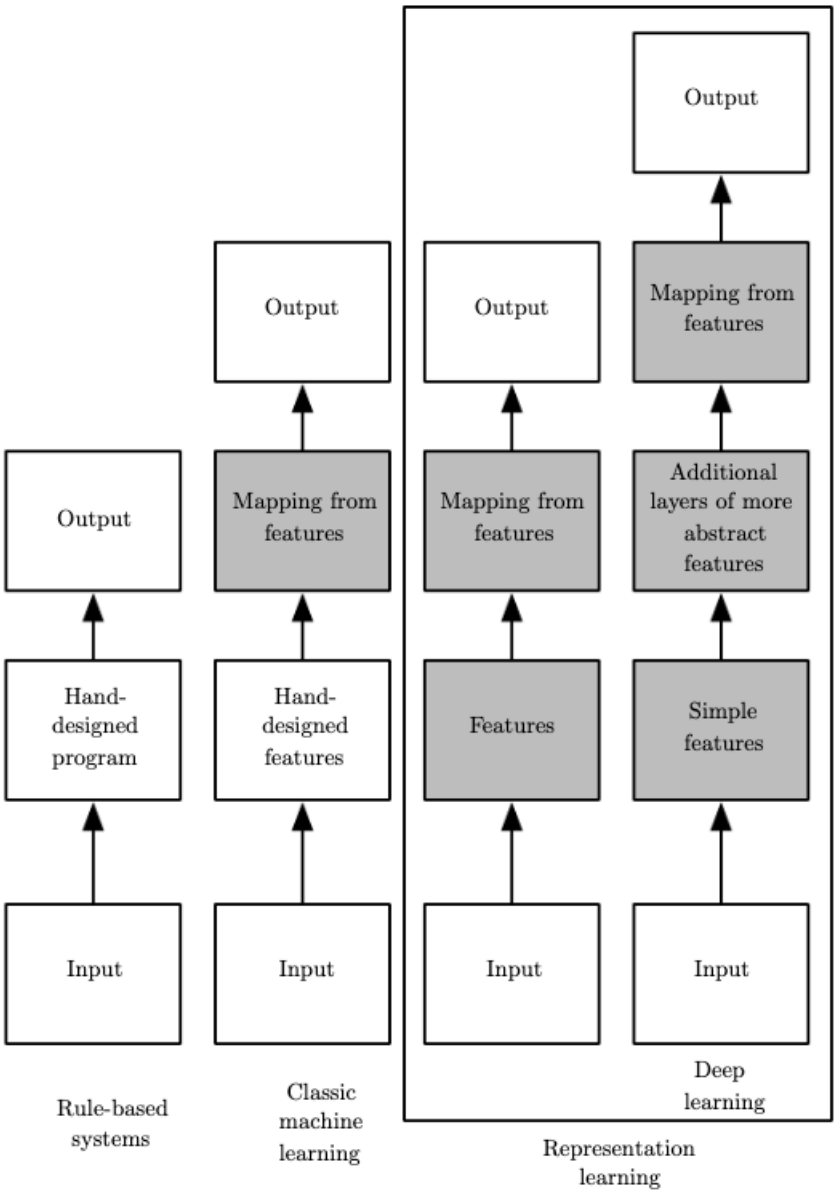


Figure 3: Comparison of Artificial Intelligence (AI) Methods The difference between Rule-Based, Classical and Representation Learning methods is shown. Figure is taken from the Deep Learning Book [26]. The white boxes indicate the tasks performed by the human and the grey boxes indicate the tasks for which the algorithm is responsible in each paradigm.

3.3 Text Classification

Written text contains a great deal of information but it is unstructured from the perspective of the computer and that makes it very hard for it to understand that information. However, it is quite useful to be able to understand language and make automated decisions based on the content of the text. It can be used, for example, to detect spam in email or to do some sentiment analysis on movie reviews automatically.

One such task based on text, is to classify a document using the text it contains. This is called text classification. To do that, first, a way is needed to extract features from the text and then use those features to classify it into predefined categories. Following sections will explore the techniques from the realm of classical machine learning that can be used for text classification.

3.3.1 Word Vectorisation

One way to extract the features from a text-based document is to use the words themselves as features. The problem, however, is that words themselves are meaningless for computers. Hence, a process is needed to quantify the words and convert them into numbers which the computer can then understand. This process is called vectorisation. This process not only quantifies the words but also builds a vocabulary using all the words from all the documents which provides a context to individual words as to where they belong. Following are two techniques for word vectorisation.

Count Vectoriser

Count Vectorisation is a bag-of-words approach where the features of the documents are the words and the frequency of occurrence of those words. The premise is that

documents that belong to the same class will contain the same words and words that are important for a class of documents will appear more frequently.

Term Frequency - Inverse Document Frequency (TF-IDF)

TF-IDF improves upon the basic count vectorisation by quantifying the importance of a word to a specific document within a given collection of documents. The basic form of tf-idf used is described below mathematically [27].

Given a term t, a document d in a set of documents D, with f representing the raw count for the term, the term-frequency is given by dividing the frequency of appearance of that term in the document with the frequency of all terms in that document:

$$tf(t,d) = \frac{f_{t,d}}{-\Sigma_{t' \in d} f_{t',d}}$$

Similarly, given a term t, a document d in a set of documents D, with N representing the total number of documents, then the inverse-term-frequency is given by:

$$idf(t,D) = \frac{N}{|d \in D : t \in d|}$$

Finally, the tf-idf term is defined by multiplying the above two terms.

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D)$$

Observing the above, it can be noted that as the frequency of occurrence of a word in a document increases, the tf value increases but idf keeps checking the occurrence of the word in the overall collection of documents and if the word appears more, generally, across all documents in the corpus then its importance for classification

purpose decreases. This helps in adjusting for the fact that some words may appear often in a given set of documents.

3.3.2 Classifiers

Classifiers are the class of algorithms that will help in classifying the text-based documents into predefined categories. The classifiers will use features from the documents to classify them. These features have been described above. In the following sections, two machine learning based classification algorithms will be discussed.

K Nearest Neighbours

K Nearest Neighbours is a simple, non-parametric and supervised machine learning model which was first developed in 1951. It can be used for the purpose of classification as well as regression. For the required tasks, it will be used for classification of text-based documents.

Whenever a new document needs to be classified, its neighbours are identified based on the similarity of its features with the similarity of the features of the documents that are present in the known dataset. The number of neighbours to be considered are K, hence the name K Nearest Neighbours. The document is classified in the class to which most of its neighbours belong.

Naive Bayes

Naive Bayes is a probabilistic supervised machine learning model that is based on Bayes Theorem. It carries with it a strong assumption that the features are independent of each other. The type of Naive Bayes that was used for our task is called Multinomial Naive Bayes because it can deal with multiple classes. With the

help of equations taken from the Information Retrieval book [28], it is described mathematically below.

Given a class as c with new document as d, the probability of the document belonging to that class is given as:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

$P(t_k|c)$ can be thought of as the probabilistic evidence that the term t_k belongs to class c. $P(c)$ is the background frequency of the class c and is interpreted as the probability of the class occuring relative to other classes. If the terms t_k do not provide enough evidence for the document to belong to any class then $P(c)$ takes over and the document is assigned to the class that has the highest background frequency. The term $P(c|d)$ for all the classes is calculated and the one with the highest value is selected. The background frequency and the probability for the terms are learned from the training dataset.

With N_c being the total number of documents in class c and N being the total number of documents, $P(c)$ is given by:

$$P(c) = \frac{N_c}{N}$$

Additionally, the term probability $P(t|c)$ is learned by calculating the relative frequency of term t belonging to class c. For the given class, the frequency of occurrence of the term is divided by frequency of occurrence of all the terms in the vocabulary:

$$P(t|c) = \frac{T_{ct}}{\Sigma_{t' \in V} T_{ct'}}$$

3.4 Deep Learning

As discussed in section 3.2.4, Deep Learning belongs to the class of Representation Learning that can learn task specific features from data using Neural Networks with 3 or more layers. The word 'Deep' in Deep Learning is related to the fact that there are a number of hidden layers (layers that are not the input or the output, essentially hidden from the user) in these Neural Networks. Following is a discussion about the origin of Neural Networks, how they learn and what types of neural networks form the backbone for the required tasks.

3.4.1 Basics

The basic component for Deep Learning is a Neural Network. As the name suggests, these networks are inspired by the human brain. Neurons are a fundamental part of our nervous system and are responsible for passing the information. They receive signals from external environment and also send information to all parts of the body for their proper functioning. They communicate with each other using electrical impulses. To understand how exactly our brain is able to create complex patterns by just connecting together this basic building block (a neuron) Warren S. McCulloch and Walter Pitts conducted their research [29] in 1943. One of the core ideas coming out of this research was the comparison of the neurons with a binary threshold to boolean logic [30].

This core idea of building a network of neurons and representing the firing or not-firing of neurons using 1s and 0s of boolean logic is the basis of all modern artificial neural networks as well. Frank Rosenblatt, in 1958, first brought this idea into the digital form when he created the Perceptron [31]. Figure 4 (a) shows the original perceptron idea that behaves as a binary classifier (i.e. depending upon the input, either classifies the input into one of the two possible categories). The figure presents the functionality of one neuron graphically. It has a number of inputs, which can be

other neurons before it, these inputs have values which are multiplied by a certain weight and then summed up before being passed through a non-linear activation function. This neuron is a core part of the neural network revolution. Figure 4 (b) shows one such neural network (called a Multiple Layer Perceptron). Each circle in this figure is a neuron with the links between the neurons carrying the weights. The modern neural networks follow the same concepts but have a great deal larger number of hidden layers (sometimes more than a 100) and a great deal larger number of neurons in those hidden layers. Mathematically each of the neuron performs the following functions.

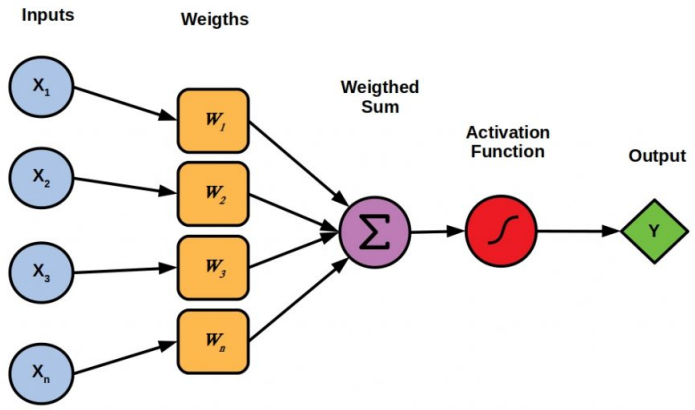
The number of weights equal to the number of inputs are multiplied with values of the inputs and a bias constant b is added.

$$z = b + w_1 x_1 + w_2 x_2 + ... + w_n x_n$$

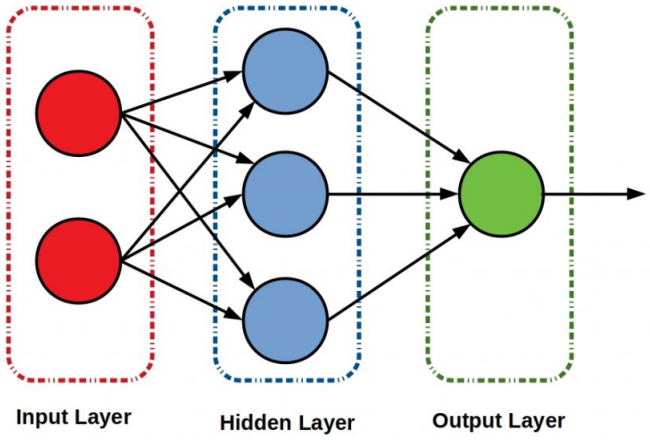
The result above is then passed through a non-linear activation function σ which then becomes the activation value of the current neuron.

$$y = \sigma(z)$$

This process goes on for all of the neurons until a final output is generated which the user receives.



(a) Perceptron



(b) Multiple Layer Perceptron

Figure 4: Perceptron Images taken from [32]

3.4.2 Learning

The neural network is defined by the number of neurons, number of layers and the weights associated within the networks. For any task, a neural network can be created but at the start the performance will be very poor because the correct weights for the task cannot be known. Hence, there is a need to adjust the weights according to this task. This adjustment of weights is called Learning. In this section, how a neural network learns will be presented.

Loss Function

As already shown, a neural network takes in an input and by successive operations and activations of neurons in the hidden layers, it produces an output. This output depends on the weights of the interconnections between the neurons. In order to adjust these weights so that they produce accurate results, a way to quantify the accuracy of the output generated by the network is needed. This quantification is done with the help of a loss function (also sometimes called the cost function).

A loss function quantifies the accuracy of the outputs by measuring the difference between the produced output by the network and the ground-truth (this ground-truth is often provided by a human). The difference between the output predicted by the network and the ground-truth value is called as loss and the goal of the learning is to reduce this loss by repeatedly showing the network known inputs and adjusting the weights accordingly.

The loss function can affect the learning process, by tweaking the loss function the learning process can be nudged in the right direction. The learning process itself is stochastic but through appropriate choice of the loss function helpful conditions can be generated so that the learning process is successful and the network starts to produce good results. One example of such a loss function that can be used for

a classification task is called Cross Entropy Loss. Its mathematical form is shown below [33].

For the case of binary classification: Assuming p as the predicted probability of the data-point belonging to class 1 and y is the ground-truth label for the class i.e. either 0 or 1, then the loss would be given as:

$$loss = -(ylog(p) + (1 - y)log(1 - p))$$

For the case of multi-class classification problem: Assuming N is the total number of classes, p_c is the predicted probability of the current class by the network and y_c is the ground-truth indicator. If this is the correct class for the current observation or not (indicated by a 0 or 1) then the loss would be given by:

$$loss = -\Sigma_{c=1}^{N} y_c log(p_c)$$

The cross entropy loss function is only shown as an example here, in practice different loss function can be used depending upon the task.

Gradient Descent

In the above section, it has been already discussed the loss function and the fact that in order to make the network better, the loss function is required to be minimised. The question arises, how to do that? The answer is gradient descent.

The input passes through the network, being multiplied with weights of the network in the process and a result is produced which gets fed into the loss function. This way, the loss function can be thought of as a function of all the weights of the network.

Taking the gradient of the loss function with respect to any of these weights in the network will tell the direction of maximum change for the function. It is, then, just needed to change the weight in the opposite direction to the calculated gradient. Doing so will reduce the loss. If this is done for all of the weights of the networks, the network performance will gradually improve.

Mathematically, if w is one of the weights in the network, L is the loss function being used, and α is the learning rate (a hyperparameter that governs how much to change the weights), then one step of gradient descent looks something like this

$$w_{updated} = w_{current} - \alpha \frac{\partial L}{\partial w_{current}}$$

Backpropagation

Now that the method of gradient descent is understood, it is needed to be able to calculate the gradient of the loss function with respect to every weight in the network. This was a hurdle in training the neural networks until the invention of the Backpropagation algorithm. The algorithm is based on the chain-rule [34] in calculus.

Consider the very simple example of the perceptron shared in 3.4.1. If the gradient of the loss function with respect to one of the weights w_1 is to be calculated, it can be done so using the chain rule as following.

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial w_1}$$

The above example is for the very simple case of a single neuron but the same methodology can be extended to multiple neurons and multiple layers by propagating the gradient from the last layer backwards.

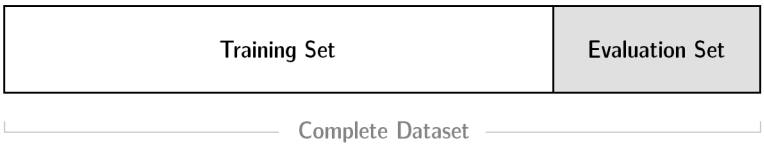


Figure 5: Splitting complete dataset between training and evaluation sets The figure shows how the full dataset can be split into training and evaluation subsets to test how well the deep learning models generalise.

Generalisation

After presenting the basics of the training/learning procedure of the neural networks, some characteristics of the learning behaviour are discussed. For the case of supervised learning, a dataset of labelled data points that are used for training the network is present. This dataset is divided into two categories called as the Training set and the Evaluation set. Usually 80% of the dataset is used for training and 20% of the dataset is used for evaluation. This is shown in figure 5.

During the process of training, the network is shown the data points from the training set repeatedly and the weights are adjusted via Gradient Descent and Backpropagation. Over time, it will be noticed that the training loss gets reduced. If this is not the case, it means that the network is not complex enough to be able to learn from the datapoints and that the network is under-fitting on the dataset.

If on the other hand, the loss during the training is being reduced but the loss on the evaluation set is increasing over time, this is the case of overfitting. This means that the network is fitting too well on the training network and as a result it is learning the wrong underlying function for the task.

In order to generalise well, it is important for the network to drop the loss on both the training set and the evaluation set.

Hyperparameters

Since deep learning is a form of representation learning, there is not much to be controlled because the network learns the parameters on its own through the process of gradient descent. However, there are still high level parameters that can be controlled that can have a great impact on the learning performance of the network. These are called Hyperparameters.

One such hyperparameter mentioned previously is the Learning Rate α. This needs to set up very carefully because if it is too low then the network will take forever to learn and if it is too high the network will change the weights too much and not learn i.e. the loss will remain high.

Other hyperparameters are the number of layers in the network and the number of neurons in those layers. If the number of layers are increased, it increases the complexity of the network and can help with the underfitting problem discussed in the above section. Similarly in the case of overfitting, the number of layers of the network can be reduced to reduce the complexity of the network.

3.4.3 Architectures

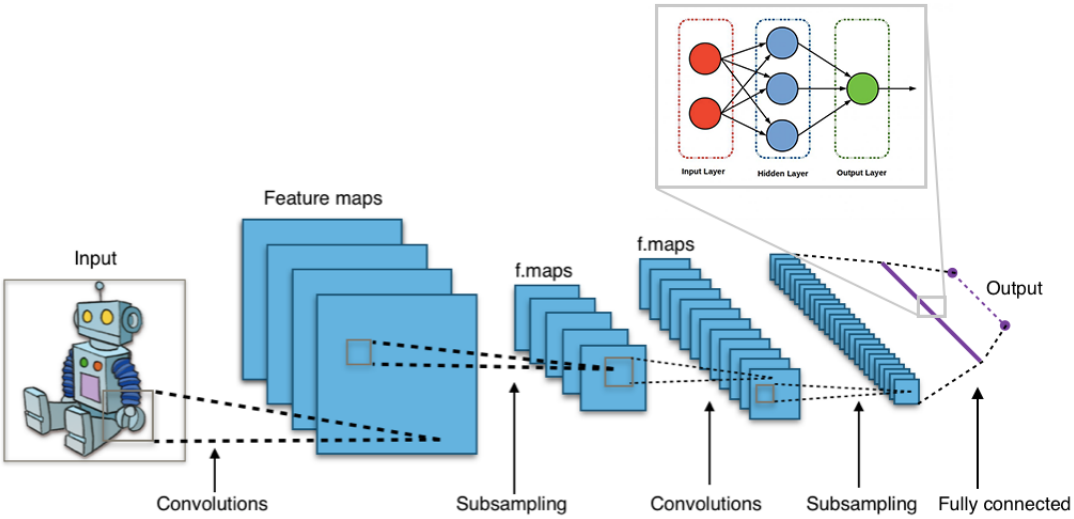
As learned previously, neural networks are formed by linking together individual neurons. The way the neurons are linked together in the form of layers is called the architecture of the neural network. There are different architectures for performing different tasks. Here, Convolutional Neural Networks and Residual Networks which are the standard for processing image based data will be discussed.

Convolutional Neural Networks

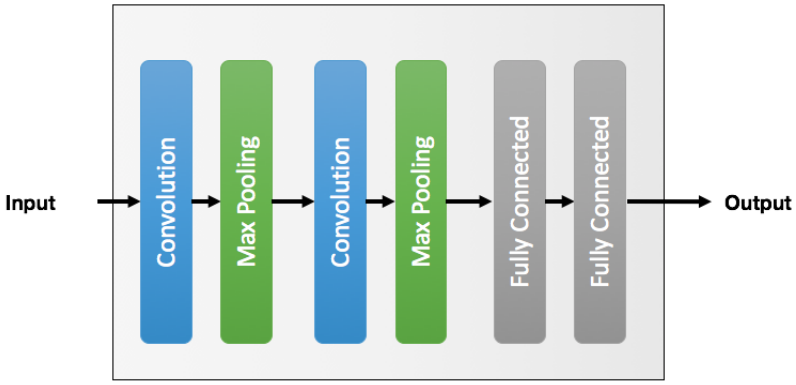
Convolutional Neural Networks (CNN) are a type of neural network architecture that are biologically inspired [35]. In 1962, Hubel and Wiesel conducted a research to study the visual cortex system of cats [36]. The study revealed the presence of what they called simple cells and complex cells. Simple cells detect certain features at certain spatial locations and complex cells receive input from these simple cells and respond to them. A hierarchy of sorts was highlighted. Later in 1980, Fukushima transformed the insight from this experiment into a model of the visual system called Neocognitron [37]. This model is the precursor to modern Convolutional Neural Networks.

In CNNs, instead of multiplication of weights in successive layers another mathematical operation called Convolution is used. Small filters are designed that are convolved over the input image. The weights of these filters are learnable. A typical CNN architecture is shown in figure 6. The receptive field becomes narrower and narrower deeper into the layers. In this way the filters at the starting layers act as simple cells and the filters at the later layers act as complex cells. The process of learning will force the network to learn the weights of the filters in such a way that filters are tuned to detect features in the image that improves the network performance overall.

CNNs can be considered a regularised version of fully connected neural networks because they have much less connections between the layers thus making them less prone to overfitting. This also reduces the computation overhead and reduces the training time. Due to the nature of the convolution operation, CNNs are scale and shift invariant. Both these properties are especially desired for image based data. The core purpose of CNNs is to extract useful 2D features from the input data that can be used for task specific applications. For the required tasks, the extracted features will be used for detection of an object (table) from the image.



(a) A typical CNN process [38] is shown. Even though the process is different than Multiple Layer Perceptrons, as shown in figure 4, in the initial layers, the final layers resemble them.



(b) A typical diagram of CNN [39] is shown. This representation of CNNs is more common in academic literature than the one depicted above.

Figure 6: Typical CNN Architecture In a typical CNN architecture, filters are applied via the operation of convolution successively. Followed by a flattening operation for the task specific output generation.

Residual Networks

When Deep Learning was growing in popularity in the domain of image recognition, there was a trend of just increasing the number of layers in the network to increase performance. This went on up to a certain limit. After a certain point, the very deep networks actually started to show worse results than the shallow networks. This was not the case of overfitting on the training set because the training loss was also not going down as much for the deep networks as compared to the shallow networks.

The breakthrough against this problem was achieved by the work of Kaiming et al. [40] where they proposed the concept of Residual Networks. The main idea was to create very deep networks by assembling together what they called "Residual Blocks". The residual blocks had skip connections for the information flow. Skip connections are additional to the existing interconnections between the blocks that skip over some blocks. As we observed in the process of backpropagation, the gradients are calculated by multiplication of a number of entities. This can lead to vanishing or exploding gradients because the value is less than 1, multiplying it over and over again will lead to a diminishing value and if the value is more than 1, then multiplying the value over and over again will lead to an exploding value. The residual connections allow a better flow of the gradients. In addition, the skip connections make it easier for the residual blocks to more easily learn the identity function (i.e. act as if the block does not exist in the network) which makes the very deep network at least perform just as well as the shallower network. If the residual block does learn something useful then you can even achieve a gain in performance.

The residual connection helped in creating very deep networks and these residual connections are standard in modern architectures these days. Such a network is shown in figure 7. These Residual Networks will form the backbone for feature extraction in the task of object (table) detection.

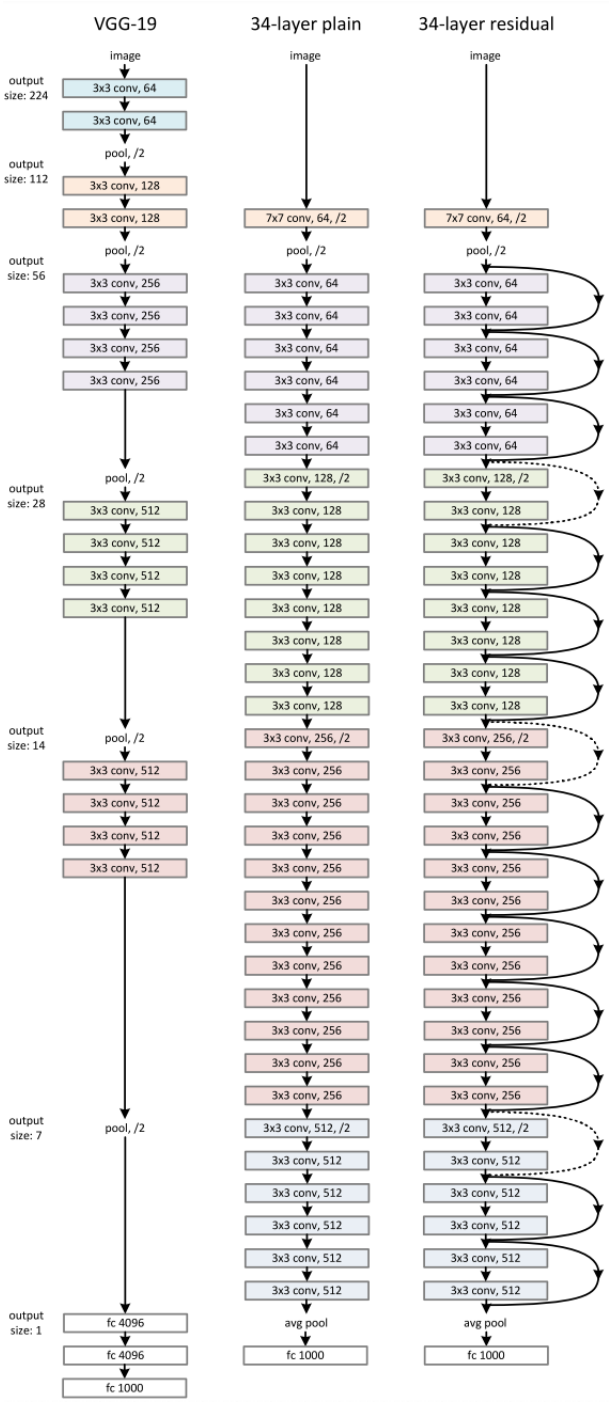


Figure 7: ResNet Architecture The ResNet architecture that was proposed compared to plain deep neural networks. The skip connections in the residual network can be observed. Figure is taken from Kaiming et al. [40].

3.5 Object Detectors

Object Detection is a field of computer vision that deals with teaching the computers to detect objects of interest in image-based data. A great deal of performance increase in the area of object detection thanks to deep learning in the recent years has been observed. Object detection has a number of applications ranging from self driving cars to automatic detection of defects in the manufacturing processes using cameras. Object detectors are interesting for the required task because much like detecting a cat or a dog in an image, we can also detect tables in the image rendered from a PDF document. Following sections will present the types of object detectors that are used for this purpose.

3.5.1 Two - Stage Object Detectors

The first of the major two types of object detectors is a Two - Stage object detector. As the name suggests, it involves two separate steps. The first step is region proposal and the second step classification. During region proposal, the network proposes a set of regions in the image that might contain the object of interest. In the second stage the proposed regions are classified into one of the pre-defined categories. Following are the two examples of two-stage detectors discussed.

FasterRCNN

RCNN (stands for Region-based Convolutional Neural Network) was a popular two-stage object proposed by Gershick et al. [41] in 2014. It combined the traditional computer vision techniques with the modern deep learning methodologies. The region proposal is done by the Selective Search methodology [42]. It proposes around 2000 regions per image. On each of these regions the CNN for classification is applied. This works but is very slow as the CNN is being applied to all of the regions individually.

A large performance gain was achieved in Fast RCNN [43] where instead of applying the CNN to all of the proposed regions, the CNN is applied to the image only once to extract the features and the regions proposed by Selective Search are projected onto these features.

However, in both RCNN and FastRCNN the selective search for region proposal was identified as the bottle neck for speed of the network. FasterRCNN was presented by Ren et al. [44] where the region proposal by Selective Search was replaced by another convolutional network which they called Region Proposal Network. This resulted in significant performance improvement in terms of speed.

The architecture for FasterCNN is presented in figure 8. The FasterRCNN that is used is also based on the work by Ren et al. where the backbone for feature extraction is a combination of Convolutional Residual Network (as discussed in section 3.4.3) along with a Feature Pyramid Network presented by Lin et al. [45]. The feature pyramid network improves the detection of objects at multiple scales [46].

FasterRCNN V2

The second type of two-stage object detector that is used is an improved version of FasterRCNN noted from the work of Li et al. [47]. The notable difference between the implementations is that FasterRCNN uses a simple two-layer Perceptron (two fully connected layers of neurons) for classification and regression of the proposal boxes whereas FasterRCNN V2 also includes Convolutional Layers in this process before the fully connected layers of neurons.

3.5.2 Single - Stage Object Detectors

The other major type of object detector are Single - Stage Object Detectors also known as Single Shot Detectors (SSDs). In this case everything is done in a single

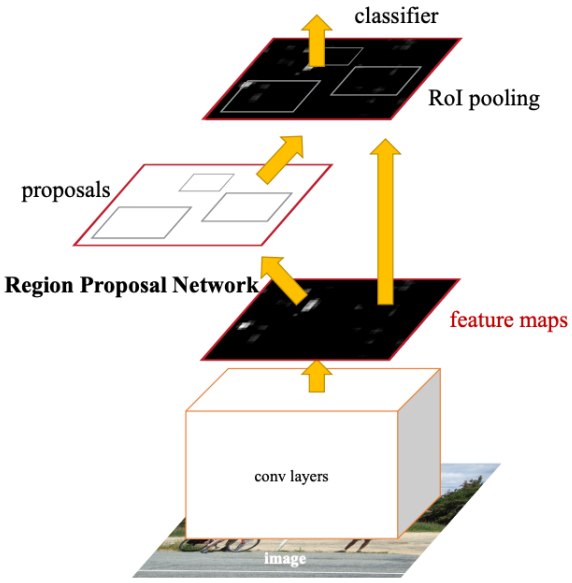


Figure 8: Faster RCNN Architecture The figure shows the high level architecture for FasterRCNN where the region proposal network is also based on Convolutional Neural Networks. Figure is taken from FasterRCNN paper [44].

step, there is no region proposal step where regions are proposed first and are then classified. The bounding boxes for the objects of interest and the classes of the objects of interest are predicted in a single step. This makes the SSDs faster than the two-stage detectors but less accurate.

RetinaNet

RetinaNet is the first of the single shot object detectors that is used. It was first presented as part of the paper Focal Loss for Dense Object Detection [46]. The paper focuses mainly on a novel loss function called Focal Loss. As single-stage detectors do not have a separate Region Proposal Network to narrow down the regions of interest, the network has to evaluate 10^4 - 10^5 regions per image. Since, the image will contain much less objects than the regions evaluated, most of these regions are easy negatives (they are not the objects of interest, thus the network should classify

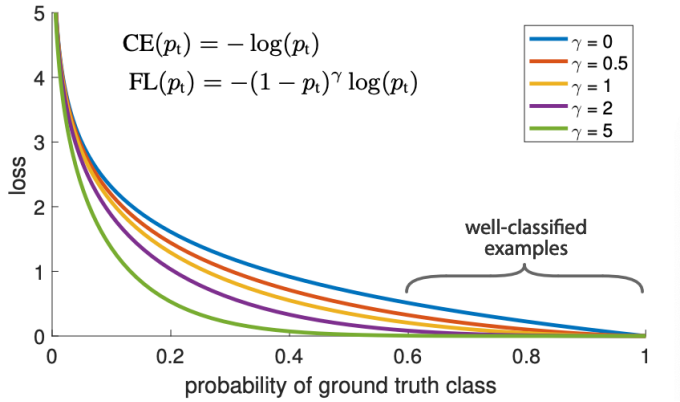
them as negative). This results in a class imbalance problem as the training becomes inefficient because the easy negatives do not contribute to useful learning and the easy negatives can also overwhelm the learning process and result in degenerate models [46]. The Focal Loss helps in this regard as it dynamically weighs down the learning from the easy to classify regions and lets the network focus more on the regions that are hard for it to learn. The characteristics of the focal loss are shown in figure 9 (a).

The figure 9 (b) shows the proposed architecture for RetinaNet. The architecture was purposefully kept simple in order to show the gains achieved by focal loss. The backbone for feature extraction is a combination of Convolutional Residual Network as discussed in section 3.4.3 alongwith a Feature Pyramid Network. The basic feature extraction is followed by two subnetworks that predict the class and regress the bounding boxes for the objects of interest in the image.

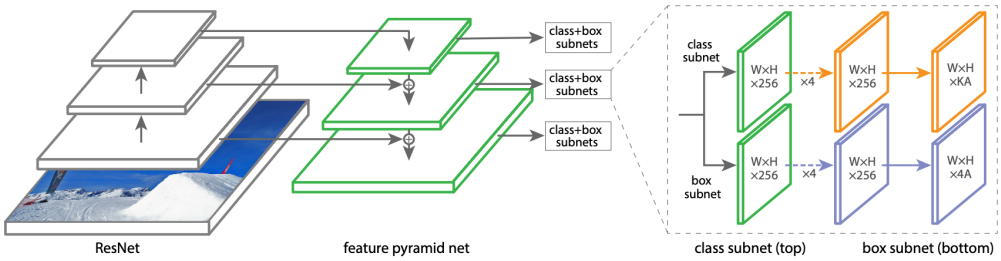
RetinaNet V2

The second single-stage detectors that we used is named RetinaNet V2 here as it improves on the original RetinaNet architecture. It uses Group Normalisation instead of Batch Normalisation in the later prediction layers of the network. Normalisation is used extensively these days in neural networks. It helps in smoothing the loss surface by applying a tighter bounds to the calculated gradients and as a result the training becomes more efficient. It also helps in keeping the contribution of all the features and with the problem of vanishing and exploding gradients [48].

One popular type of normalisation is Batch Normalisation. As the name suggests, it normalises the features across a batch but it will only work well if the batch size is large which is not always possible due to limitations of memory or other reasons. For this purpose an alternative known as Group Normalisation is used. Group Normalisation is independent of the batch size and it is consistently better than



(a) Loss Function Characteristics



(b) RetinaNet Architecture

Figure 9: RetinaNet Architecture and Loss Function Images taken from [46]

Batch Normalisation at lower batch sizes [49]. As will be noted later, the batch sizes used in our experiments are also quite low. Thus this improvement in architecture is helpful for us.

Another improvement made here over the original RetinaNet is the implementation of Generalised IoU loss for the regression of bounding boxes. IoU is a popular metric (also the one that is used for evaluation of models later) for measuring the accuracy of bounding boxes that are generated for the object of interest. In the RetinaNet, simple type of distance-based (it calculates the distance between the predicted values and the ground-truth values) loss function known as L1 loss is used for regressing the coordinates of the bounding boxes. However, there is a gap between optimising the commonly used distance losses (such as L1 loss) for regressing the parameters of a bounding box and maximising this metric (IoU) value. The optimal objective

for a metric is the metric itself [50]. Both of these improvements, implementation of Group Normalisation and Generalised IoU loss, are taken from the work of Zhang et al. [51].

4 Approach

In chapter 3, the background of all the technologies that are used to solve the problem of extracting information from PDF data sheets was discussed. In this chapter, the approach that we took in order to achieve the desired results will be. Before going deep into the detail of the methodology, the problem that should be solved must be clearly defined.

4.1 Overview

In view of the concerns mentioned in section 1.2, the problem of extracting information from PDF files can be divided into three major steps. These are shown in figure 10 and figure 11. The system takes in the PDF document and detects the location of the tables in the PDF documents in the first step which is called Table Detection. In the second step, Table Recognition, the raw values from the table are extracted using the detected location of the tables. In the final step, the raw values are processed, brought to structure and saved in a database. In the following sections, these three major steps are expanded upon in more detail.

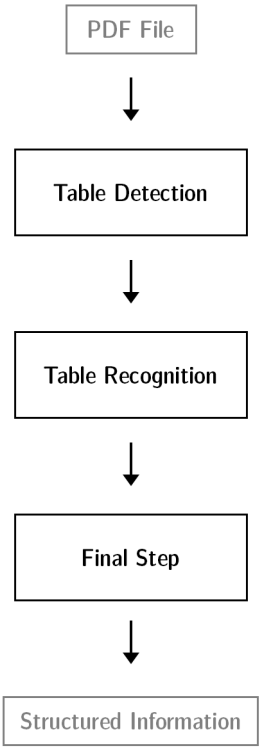


Figure 10: Overview of Approach The figure that shows the overview of the approach being taken to extract information in tables from PDF documents. There are in total three steps. Table Detection will be the process to locate tables in the documents. Table Recognition will be the process to locate the rows and columns in the tables and extracting raw information from them. The Final Step will deal with locating the required information, extracting it and validating it.

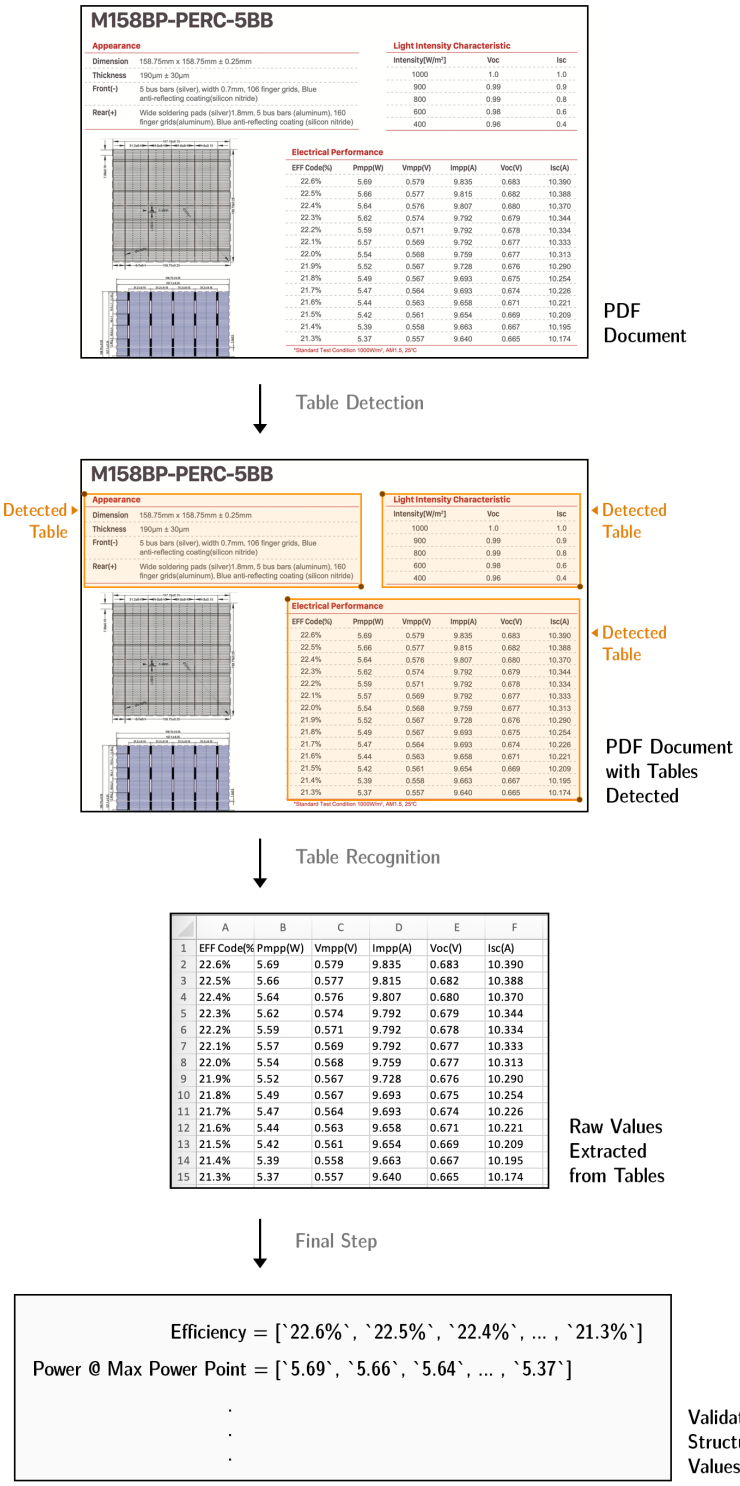


Figure 11: Overview of Approach Example Firstly, tables are detected as indicated by orange bounding boxes. Secondly, raw values of the table are extracted. Finally, required values are structured and validated. Here, the column 'EFF Code(%)' is mapped to 'Efficiency' and the column 'Pmpp(W)' is mapped to 'Power @ Max Power Point'.

4.2 Table Detection

The first of the three major steps in extracting information from tables of PDF documents is Table Detection. This, arguably, is the most important step in the pipeline since without knowing the location of the tables on the page, no information can be reliably extracted. The object detector deep learning models that were discussed in section 3.5 are used for table detection. The inspiration for using object detectors came from analysing how humans are able to spot the tables in the documents. We have a very visual process whereby we focus on the edges of the tables which are either formed by colour or hard lines. We also look for separation between rows and columns and then decide the area of the document that contains the tables.

In recent years, the field of computer vision has been revolutionised by the introduction of convolutional neural networks that are able to process image-based data and learn visual features. The motivation was to make the object detectors learn to spot the tables the same way that humans do i.e. looking at the visual features of the image. Thus, it was decided explore the area of object detectors for detecting the tables in the documents. Figure 12 shows the block diagram of the table detection approach with figure 13 showing an example with approach applied.

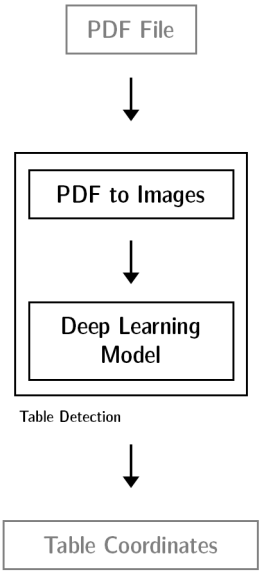
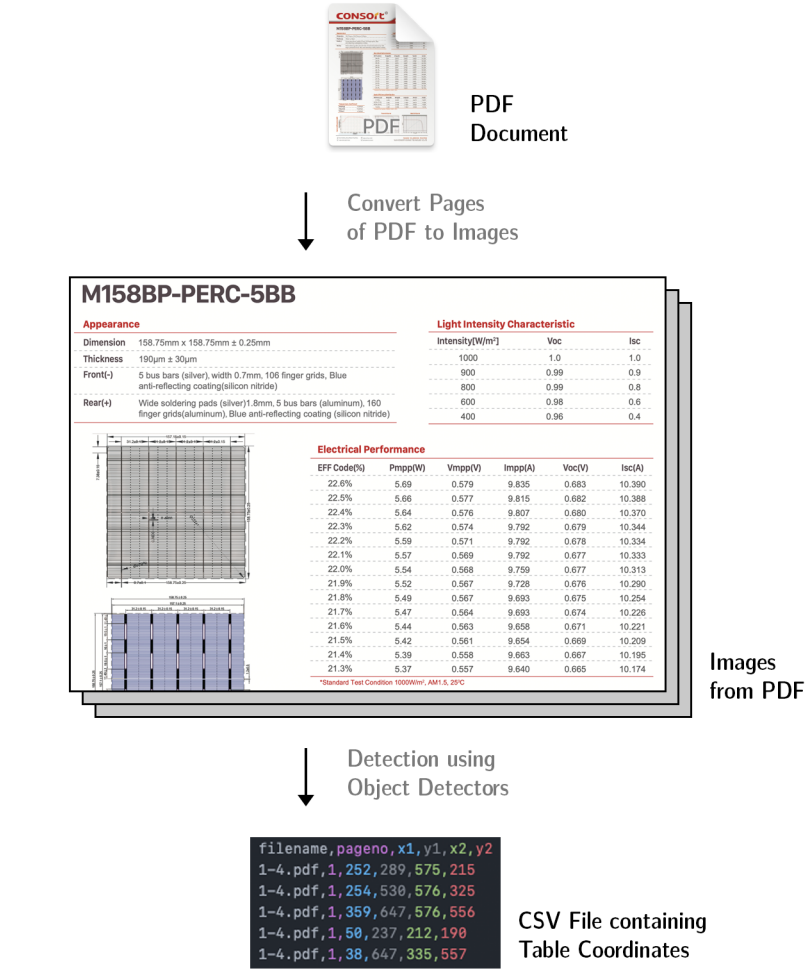
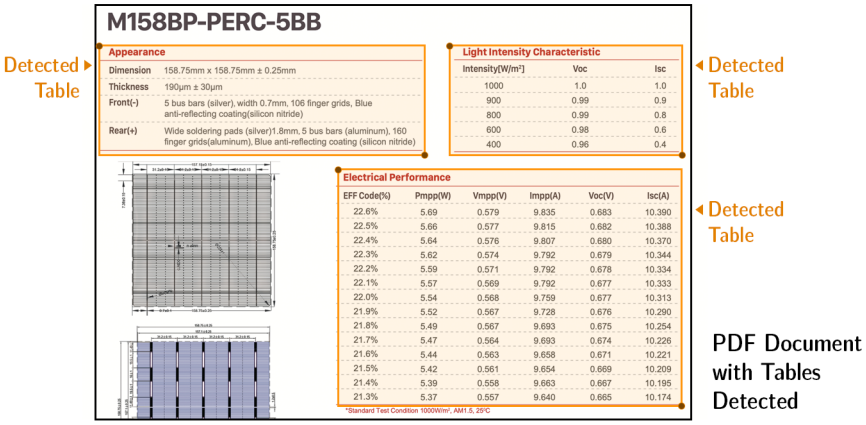


Figure 12: Table Detection Approach The figure shows the high level approach that is used for detecting the tables in a PDF document. It takes in the PDF documents, the individual pages from the document are rendered into images using a Python based library. These images are then fed into the Deep Learning model that has been trained to locate tables in the images of documents. The output is a file (CSV) that contains the information about the page and the coordinates where the tables are located.



(a) Table Detection Process



(b) Visualiation of Detected Coordinates

Figure 13: Table Detection Visualised Figure (a) shows an example for the table detection approach. Figure (b) visualises the detected coordinates.

4.2.1 Conversion to Images

As already known about the construction of the PDF documents, they have layers that contain different form of data i.e. textual, image and vector layers but the object detectors can only work with image-based data. Hence it is first required to render the PDF documents to images so that the object detectors can take them as inputs. For this purpose the PyMuPDF open source library for Python was utilised. As the PDF file is fed for the step of table detection, all the pages in the document are first rendered into images. The process also tags the images with the page numbers in order to keep track of where they come from.

4.2.2 Detection of Tables

Once the images from the PDF have been generated, they are passed through the trained object detector model. The output is the prediction of the location of the tables. The coordinates are in the form of image coordinate system i.e. top-left is the origin (0, 0). These coordinates are combined with the page numbers to create a CSV file that contains the information about the locations and the page numbers of the tables.

Any object detector model will not be able to locate the tables in the documents, they first need to be trained. To do so, sample images were manually curated from the documents and labelled for the location of tables in them. This formed the training and evaluation dataset. More information about the dataset and the object detectors that were compared are provided in Section 5.2.

The two most popular deep learning libraries that are available as of writing this thesis are Tensorflow and PyTorch. PyTorch was chosen for the required tasks due to its ease of use and community support. Furthermore, it contained an extensive library of pre-trained models that are designed and tested by the PyTorch team.

This helped in quickly developing the application without running into software engineering issues.

4.3 Table Recognition

After the Table Detection step, the coordinates of the location of the tables became known. The next step is to use these coordinates in order to extract all the raw values from those tables. This step is called Table Recognition i.e. it recognises different features of the table such as rows and columns. In this step, the focus was not what values to filter out, but rather at extracting all of the values. The Final Step can then bring structure to these raw values.

Figure 14 shows the block diagram of the table recognition approach and figure 15 shows the approach visually applied. The inputs for this step are the table coordinates that were extracted in the table detection phase and the PDF file themselves. As discussed in section 1.2, there are open source libraries available for python that aim at extracting the table data from PDF files but they perform poorly for the detection of table locations in data sheets. However, these utilities do offer the support of specifying the table locations after which they only become responsible for table recognition. Two most popular utilities, Tabula and Camelot, were compared for this purpose against a simple baseline algorithm.

4.3.1 Baseline

A simple table recognition algorithm was developed in order to provide a baseline for the comparison of two popular table extraction utilities, Tabula and Camelot. PyMuPDF was utilised to extract texts and their location information from the PDF files. The same utility was used in section 4.2 as well for rendering the PDF documents to images. PyMuPDF does not have any table recognition capabilities

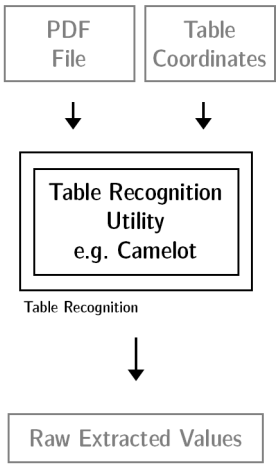


Figure 14: Table Recognition Approach This figure shows the high level approach used for the process of table recognition. In the previous step of table detection the coordinates for the location of tables were already extracted. Here the Python-based libraries e.g. Camelot can be fed the PDF documents and the information about the location of the tables and it will produce an output file (Excel) that contains the raw values that have been extracted from the tables.

itself but the information extracted using it can be used to construct a rudimentary algorithm.

Since the locations of the table were already known from the previous step, only the texts contained within those bounds are taken into consideration. The lines of text having the same vertical position are detected as rows of the table. For each row (line) the text is split on white spaces. Since the columns will have large number of white spaces between them, this split resulted in individual columns for the given row.

These extracted rows and column values were then saved in the form of Excel files where one sheet of the excel file contained one table found in the PDF file. Since this approach to table recognition is fairly basic, it will act as a baseline for the more sophisticated utilities mentioned in the following sections.

4.3.2 Tabula

Tabula is one of the two popular table data extraction utilities that is available for Python. The original Tabula project was developed in Java to suit the needs of investigative journalists. An additional open-source project tabula-py was developed which provides a Python wrapper (allowed the use of application features programmatically from Python rather than Java) for the Java based application.

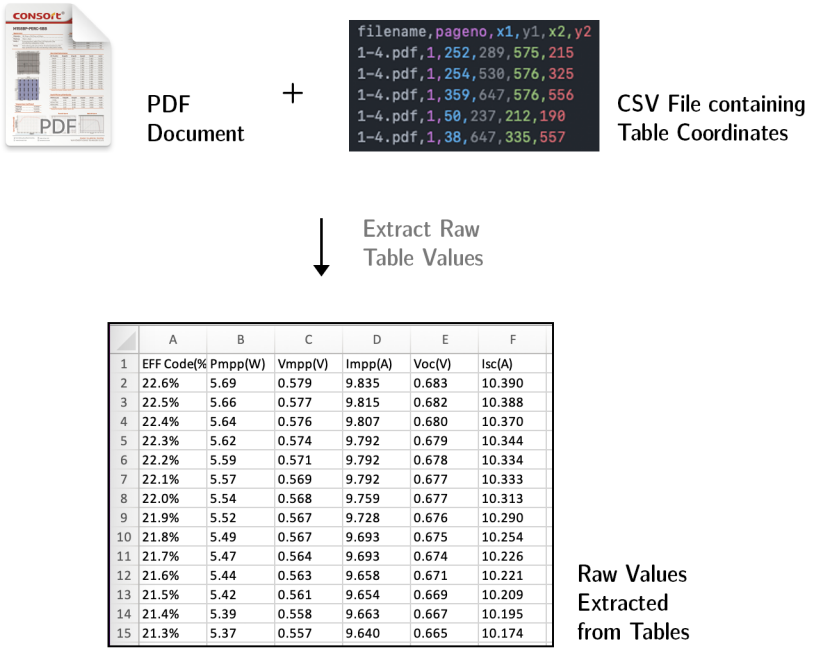
The library can be used to read the PDF file, automatically detect the tables, extract the information and save them as a CSV file. Only the table recognition capabilities of Tabula will be used as the table detection is being done using deep learning based object detectors.

To recognise the tables, Tabula reads all text contained in the table areas. For the entire height of the table, vertical areas are identified that contain no text. These empty vertical areas form the boundaries for the columns. Afterwards, based on the unique vertical positions, the rows are identified [52]. Using this approach, the row and column spans of the table are identified and as a result, the table values can be extracted.

4.3.3 Camelot

The second of the utilities is named Camelot and it is based on the master thesis work by Anssi Nurminen [9]. Like Tabula, Camelot offers both table detection and table recognition utilities. However, only the table recognition capabilities of Camelot will be used since table detection is being done using deep learning based object detection.

To recognise the table, it first assigns each table element to a row based on its vertical position. The words assigned to the same row are merged together in a sentence. Based on the row element edges, obvious column areas are identified. Elements of



	A	B	C	D	E	F
1	EFF Code(%	Pmpp(W)	Vmpp(V)	Impp(A)	Voc(V)	Isc(A)
2	22.6%	5.69	0.579	9.835	0.683	10.390
3	22.5%	5.66	0.577	9.815	0.682	10.388
4	22.4%	5.64	0.576	9.807	0.680	10.370
5	22.3%	5.62	0.574	9.792	0.679	10.344
6	22.2%	5.59	0.571	9.792	0.678	10.334
7	22.1%	5.57	0.569	9.792	0.677	10.333
8	22.0%	5.54	0.568	9.759	0.677	10.313
9	21.9%	5.52	0.567	9.728	0.676	10.290
10	21.8%	5.49	0.567	9.693	0.675	10.254
11	21.7%	5.47	0.564	9.693	0.674	10.226
12	21.6%	5.44	0.563	9.658	0.671	10.221
13	21.5%	5.42	0.561	9.654	0.669	10.209
14	21.4%	5.39	0.558	9.663	0.667	10.195
15	21.3%	5.37	0.557	9.640	0.665	10.174

Figure 15: Table Recognition Approach Example An example of the table recognition approach is shown here.

the tables are then assigned to columns after filtering out the rows that break the table structure such as title and header name rows. Finally, a clean up operation is performed that merges the incorrectly separated columns and rows [9].

Similar to Tabula, Camelot also can save the recognised tables in different file formats. In all of the cases (baseline, tabula and camelot), the raw values extracted from the tables are saved excel files. The results from the comparison of the three approaches are mentioned in the Chapter 5.

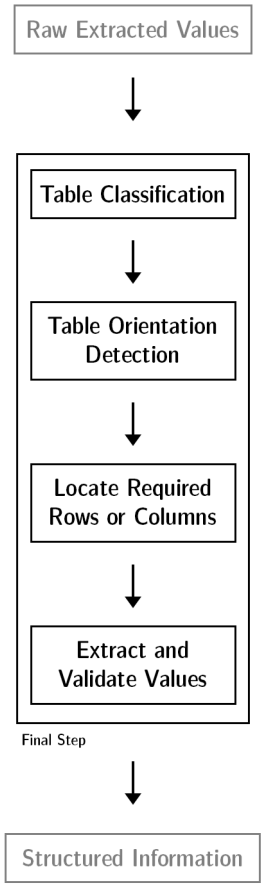


Figure 16: Final Step Approach The figure shows the high level approach for that is taken as the final step in processing the raw values that are extracted from the table recognition process. The first step is to determine the correct class of the table, after which table specific rule-based algorithms can be used for extracting and validating values from rows or columns. The table orientation detects if the table is vertical or horizontal.

4.4 Final Step

The goal of the previous two steps was to locate the tables and extract the raw values from them. However, there is still a need to process the raw values in order to filter out the values of interest from all the tables found in the data sheets. This process is called 'Final Step'. Figure 16 shows the individual steps involved in the process with figure 17 showing the results of the steps on an example table. They will be expanded upon in the following sections.

4.4.1 Table Classification

In the final step of the pipeline, each type of tables has its own rules for correct extraction of values within. For this reason, the type (class) of the table needs to be identified from its raw values. The raw values are extracted from all the tables that are found in the data sheet. In some cases the tables do not include their titles. In addition, the location of where the title appears in the table cannot be specified with certainty. Thus, it is very hard to identify the type of the table by tracking the title. In order to solve this problem, it was decided to use the machine learning techniques mentioned in the section 3.3. This can be done because of the textual nature of column and row names in the tables. Figure 18 shows an overview of the approach.

Since for this purpose, the techniques from the realm of supervised machine learning were utilised, a dataset was required that can be used for training and evaluation purposes. In order to create diversity in the dataset to capture the various ways manufacturers name the tables and the content within it, 60 excel files that contained the raw values from the tables were randomly selected. The tables within these excel files were labelled manually. The word vectorisation techniques that were used are already described in section 3.3.1. The implementations for these were used from the very popular open-source package for Python named Sci-kit learn.

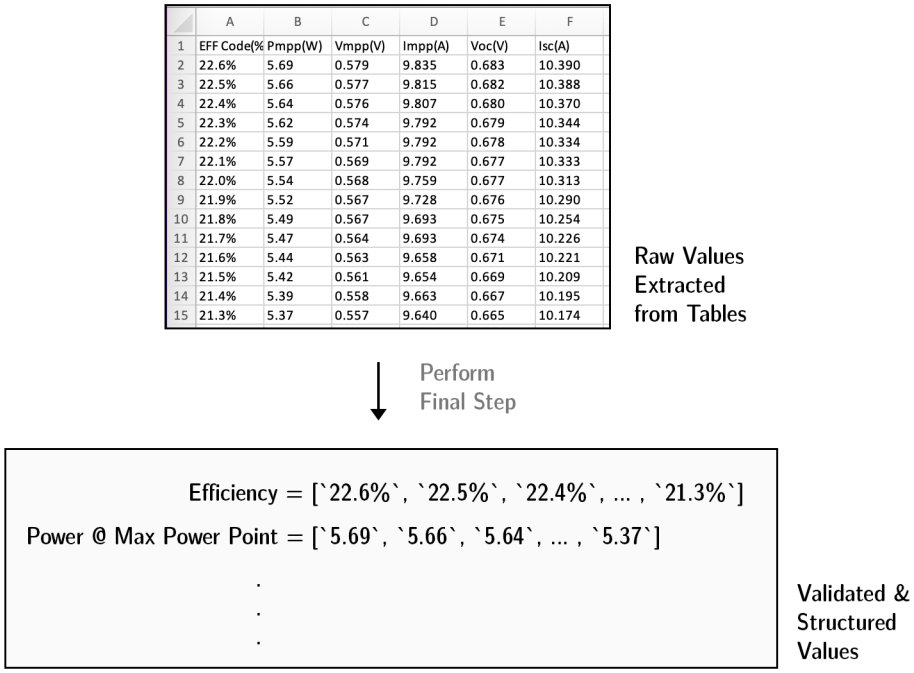


Figure 17: Final Step Approach Example An example of the final step approach is shown here. The required values are structured and validated. Here, the column 'EFF Code(%)' is mapped to 'Efficiency' and the column 'Pmpp(W)' is mapped to 'Power @ Max Power Point'.

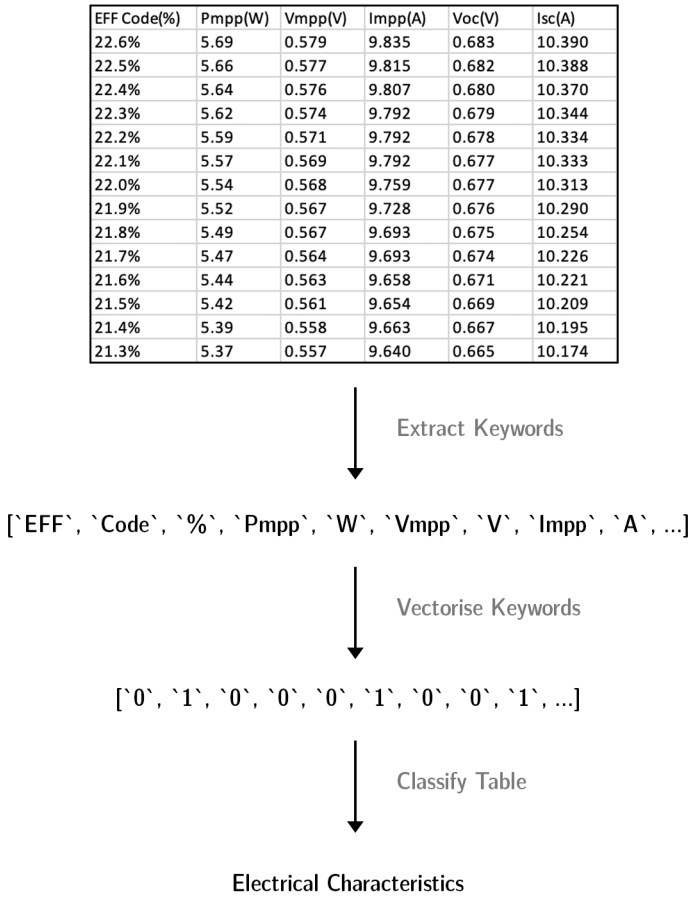


Figure 18: Table Classification Procedure This figure is a pictorial representation of the table classification approach. First, the keywords from the table are extracted. These keywords are then passed through a word vectoriser that puts them into context in accordance with the whole word vocabulary. These vectorised keywords are then used by the classifier to make prediction of the class of the table.

In order to use the package Sci-kit learn, the text in the table needed to be transformed as Sci-kit learn cannot read tabular data by default. The preprocessing is done cell by cell, taking in the present information. All text in the cell is split on white spaces. Afterwards, all of the numbers are removed as they do not form very good features for classification directly. This results in a list of words per table that gets assigned to the class. This lift of words can then be fed into the word vectorisation algorithm to establish a vocabulary of features necessary for classification.

The classifiers that were used are also defined already in the section 3.3.2. The implementations of these classifiers were also used from the package Sci-kit learn. Since both the word vectorisers and the classifiers belonged to the same package, they were compatible. This ease of use made it possible to compare different combinations of classifiers and word vectorisers. The best performing combination was selected. The results of this comparison are mentioned in the Experiments chapter.

4.4.2 Table Orientation Detection

After the appropriate class of the table has been determined, the second step is to determine whether the table is vertical or horizontal i.e. whether the same type of values are mentioned in the rows or the columns. To detect this, all of the numerical values (numbers excluding text) are extracted into a separate matrix. For this matrix, the variance for each row and each column is calculated separately and summed up. This is mathematically expressed below, here σ_{rn} is the variance of nth row and σ_{cn} is the variance of nth column.

$$\sigma_{rows} = \sigma_{r1} + \sigma_{r2} + ... + \sigma_{rn}$$

$$\sigma_{columns} = \sigma_{c1} + \sigma_{c2} + ... + \sigma_{cn}$$

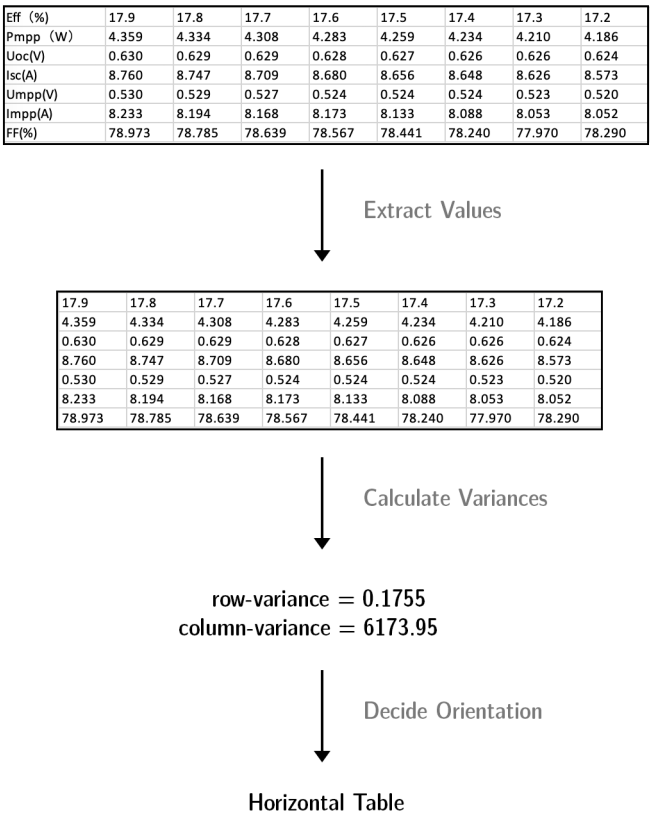


Figure 19: Table Orientation Detection Procedure This figure is a pictorial representation of the table orientation detection approach. First, the values are extracted. The row and column variance is calculated from these values. The orientation is decided based on the lower variance.

Which ever of these summed variance is lower will indicate the orientation of the table since the variation of similar values will be much lower than non-similar values. If $\sigma_{rows} > \sigma_{columns}$ then the table is vertical otherwise the table is horizontal. An example of the approach is shown in figure 19.

4.4.3 Locating Required Rows or Columns

The first two steps, the identification of type and the orientation of the table are generic and are applied the same way for all the tables. The next step is to locate the rows (in case of a horizontal table) or the columns (in case of a vertical table) that contain the information that we are interested in. Since manufacturers can have variability in how they name the column or row names and what information they include in the header of the columns or rows, it is important to deal with information split across different cells.

In order to locate the rows or columns of interest, regex pattern matching was used. Regex stands for Regular Expressions. They are sequence of characters that can help match and locate textual data. For example, instead of specifying all possible types of numbers to match only $d+$ can be specified and it will locate a sequence of numbers in the given text. All of the cells in a particular row or column are scanned and they are matched with a prebuilt repository of regex patterns for values of interest. If there is a match, the column or row is selected for further processing otherwise it is rejected. Since each table has different type of values, separate row or column identification steps are designed for each table type. One example for this process is shown in figure 20.

4.4.4 Extract and Validate Values

The last step in the process is to finally extract the values from the columns or rows that have been identified in the previous step. Since there is some variation of how

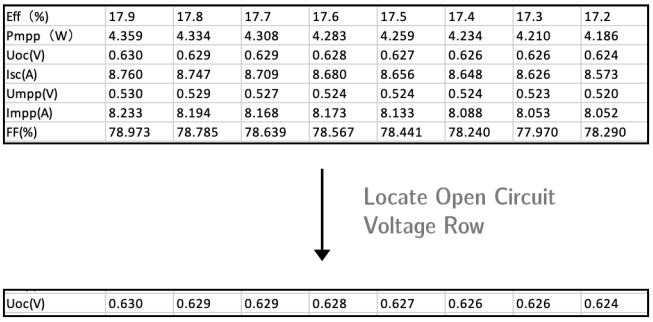


Figure 20: Row Location As we already know the orientation of the table from the previous step. We can locate the required rows using regex pattern matching. One example of a property type is shown here.

the values are written as well, some regex patterns were designed that can identify the numerical values in a particular row or column.

The values are only extracted if they match the pre-defined regex patterns for the value type. This approach also helps in resolving some of the errors that might be produced during the table recognition step such as incorrectly merged or empty cells. This is shown in figure 21.

Overall, the processing done in the 'Final Step' not only tries to cover for the variability in data representation from the manufacturers but also tries as best as possible to cover for the deficiencies in the previous steps of Table Detection and Table Recognition.

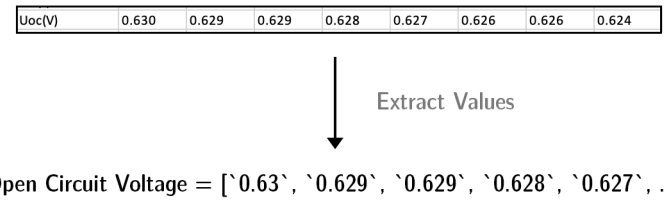


Figure 21: Value Extraction From the located row, just the values can be extracted using regex pattern matching.

5 Experiments

This chapter will present the results from 3 experiments that evaluate the critical steps in the complete approach explained in chapter 4. The results generated show the feasibility and the viability of different steps involved in the complete pipeline. In the first section, the background information for evaluation metrics that will be used for the subsequent three experiments will be presented.

5.1 Evaluation Metrics

Confusion Matrix

During evaluation, the model or pipeline is asked to predict examples that were not included in the training set. Based on the performance of the model, following values are counted.

- **TP** stands for True Positives. For a given class, these are the number of examples that were positive and the model also classified them as positive.

- **FP** stands for False Positives. For a given class, these are the number of examples that did not belong to the current class but the model falsely classified them to be a member of it.

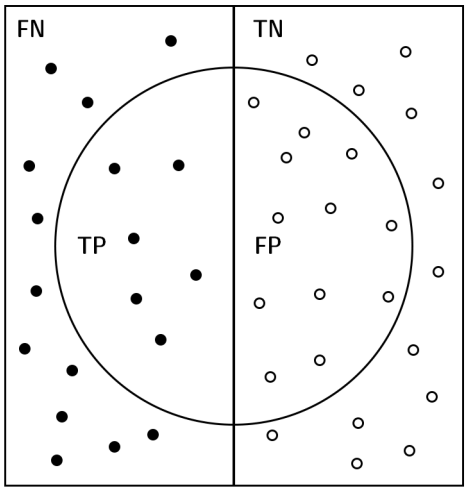


Figure 22: Confusion Matrix Visualisation The bigger box contains all the examples, both negative and positive, for a given class. The left half contains the positive and the right half contains the negative examples. The circle denotes the examples that were predicted as positive by the model.

- **FN** stands for False Negatives. For a given class, these are the number of examples that belonged to the current class but were not predicted by the model to belong to it.

- **TN** stands for True Negatives. For a given class, these are the number of examples that did not belong to it and the model also did not classify them as belonging to it.

These terms are also referred to the values of the confusion matrix. Figure 22 shows a visual representation of this confusion matrix. With the knowledge of these terms, we can now understand the evaluation metrics.

Accuracy

One of the basic metrics that can be used to evaluate the performance of a model or an approach is accuracy, which is mathematically expressed below.

$$accuracy = \frac{Total\ Correct\ Guesses}{Total\ Guesses} = \frac{TP + TN}{TP + TN + FP + FN}$$

Care must be taken when accuracy is being used to evaluate model performance as it might not provide the overall picture in case of class imbalance. That is, if the model gets very good at predicting one of classes and there are more examples in the evaluation dataset for that class, the accuracy will be very high even if the model is performing poorly on other classes.

Precision

In simple words, precision can be described as a measure of how accurate are the predictions of a model for a particular class. Mathematically, it is be expressed as:

$$precision = \frac{TP}{TP + FP}$$

Optimising for precision will make the model more cautious in making a positive prediction for a class. This metric can be cheated by making as few true positive predictions as possible. This, however, does not mean the model is good because it will fail to detect all of the examples from a particular class.

Recall

In simple words, for a given class, recall is a measure of how many of the examples were successfully detected by the model. Mathematically, it can be expressed as:

$$recall = \frac{TP}{TP + FN}$$

This metric can be cheated by classifying everything as positive. In this case, the recall score will be high but so will be the false positives which is not desirable. It might be evident that this metric is the counterpart of Precision.

F1-Score

The metrics discussed above, precision and recall, are counterparts of each other. The disadvantages of both these metrics are reduced by combining them together into a metric known as F1-Score. It is the harmonic mean between Precision and Recall. Mathematically, F1-Score can be expressed as:

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

While F1-Score can be a good way to gain a big picture of the performance of a model or pipeline, it might still be worthwhile to consider metrics described above depending upon the use-case.

IoU Overlap

IoU overlap is an important metric for object detection applications. The object detectors produce bounding boxes for the detection instances of the objects in the images. To evaluate how well, the object detector has performed there is a need to compute the overlap between predicted bounding boxes and the ground-truth bounding boxes. This can be achieved by calculated a ratio known as IoU Overlap. It stands for intersection over union. In two dimensions, for two overlapping boxes,

the IoU is calculated by dividing the area of intersection of the overlap by the area of union of the overlap.

In order to determine if a certain object in the image has been detected or not, a threshold value for IoU overlap can be specified. If the IoU overlap of the bounding box predicted by the model with the ground-truth bounding box is higher than the specified threshold, then the object is considered to be detected otherwise not. Using this, we can calculate the confusion matrix values as described in section 5.1 which can subsequently be used for the calculation of precision, recall, f1-score and accuracy.

5.2 Experiment - Table Detection

The approach for detecting tables in the PDF documents was discussed in detail in section 4.2. This section will provide the results from the evaluation of that approach.

5.2.1 Evaluation Metrics

This section will define the metrics that were used for evaluating the deep learning based object detection models for table detection.

- **Recall**: This metric has been mentioned already in section 5.1. Since for this experiment there is only one class i.e. the table, this metric can be interpreted as what percentage of the tables present in the PDF documents were detected by the model. The prerequisite to the calculation of the precision metric is IoU threshold. If the IoU threshold is relaxed then more errors from the network or the model will be tolerated and the score for recall will increase and vice versa. For the comparison of different models the Recall performance at 90%

IoU overlap was selected. In the following sections, for the sake of brevity this will be referred to as Recall-90.

- **Precision**: This metric has also been previously discussed in section 5.1. In simple language, this metric can be interpreted as what percentage of the guesses made by the network were correct. Similar to recall, the prerequisite to calculating precision is the selection of the IoU threshold. Mainly, the precision score at 90% IoU overlap will considered for discussion of results and will be referred to as Precision-90 for the sake of brevity.

5.2.2 Experiment Setup

Dataset

In order to make the dataset diverse and include the data sheets from a variety of manufacturers, the data sheets for training and evaluation dataset were sampled randomly from a bigger pool of files. These PDF files were then converted into images using the open-source software PyMuPDF in Python. This resulted in the training dataset comprising of 2675 images. The total number of tables were 5896 because most of the images contained multiple tables.

The dataset was then labelled using the open-source software LabelImg. During the labelling process the bounding boxes were drawn manually around the table regions in the images. Due to this manual procedure, there were slight variations while indicating the boundaries of the tables. In addition, because of variations in how manufacturers design the data sheets, the title of the table sometimes appeared separately from the table. A judgement call was made on case-by-case basis whether the title of the table should be included in the boundary of the table or not. The title was omitted from the table region if it appeared far away from the table.

Comparisons

The model architectures discussed in the section 3.5 were compared against each other using the Recall-90 metric. For each of these architectures, two hyper-parameters were also tuned. These hyper-parameters were the batch size (how many images are used for a single step of gradient descent) and the learning rate for gradient descent. All the networks and the hyper-parameter combinations were trained for 50 epochs. By limiting all training runs to 50 epochs, the models and hyper-parameter settings were compared more fairly since variability in epochs trained can result in under and overfitting.

In order to compare across different model architectures, it was important to fix the dataset and the division of the dataset between the training and evaluation examples. This was done programmatically by fixing the random process used for separating the dataset into training and evaluation set. This ensured that always the same files were assigned to the training and evaluation sets. For all cases, 80% of the dataset was used for training and 20% of the dataset was used for the evaluation.

The weights of all the models used in the experiments were not randomly initialised rather the weights trained on the Microsoft COCO dataset [53] were used. This allowed the network to have a good starting point and increased the training efficiency.

Training

Deep learning models have a lot of trainable parameters and multiple iterations over the training dataset need to be performed to achieve good results. This makes the training process computationally intensive and time consuming. The deep learning frameworks that is used (PyTorch) has support to accelerate training using Graphical Processing Units (GPUs) to speed up the training process.

There are a number of online services that offer GPU services. Kaggle is one such service that allows training of deep learning networks on GPUs. P100 GPU from NVidia was utilised for training on Kaggle. A single 50 epoch training run lasted 8 hours on average. One of the limitations of Kaggle was maximum training duration of 12 hours. Since we aimed to train the models for only 50 epochs, the 12 hour duration was acceptable.

5.2.3 Results

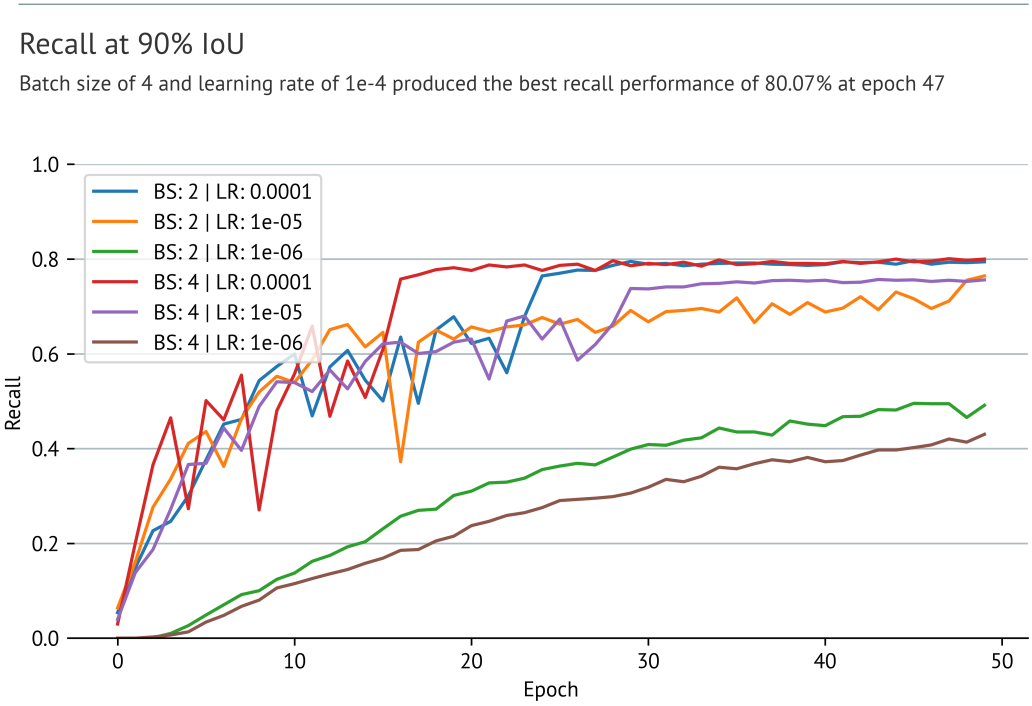
In this section, first, the results from hyper-parameter tuning for each model architecture will be presented. Afterwards, the models with the best setting of hyper-parameters will be compared to find the best one. All training runs are evaluated on the Recall-90 metric.

RetinaNet

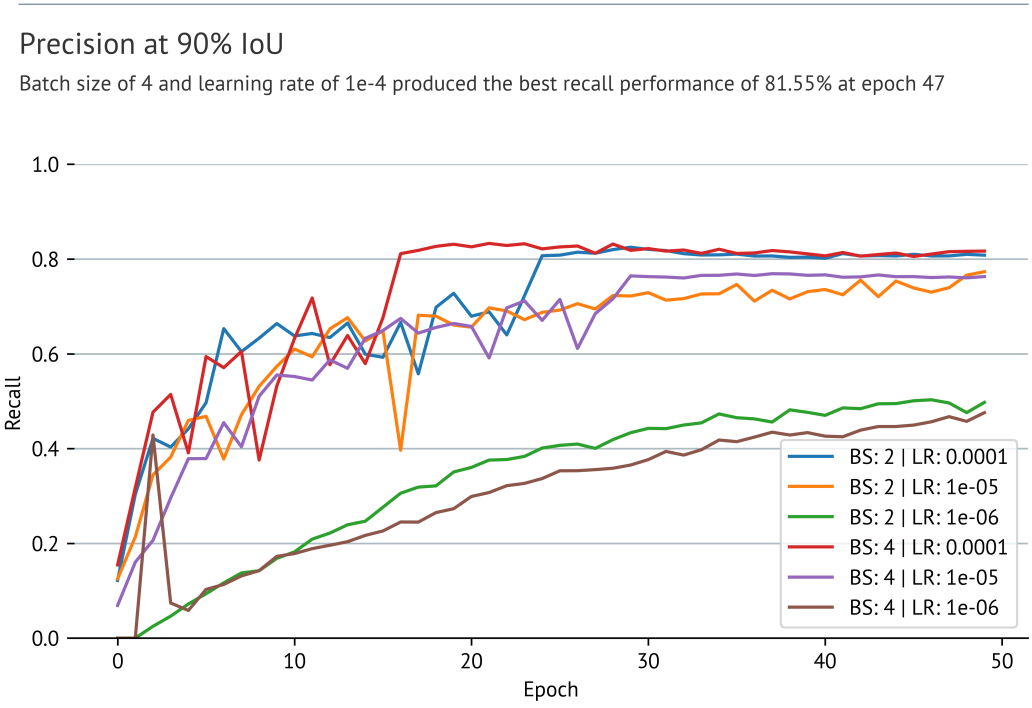
The RetinaNet was the first of the two single stage detectors. The main feature for this model was its novel loss function that claimed to improve the training efficiency of the single stage detectors. Table 1 shows the results achieved by using different batch sizes and learning rates. The best model was selected based on the best Recall-90 performance, which is highlighted in bold in the table. Figure 23 shows the Recall and Precision curves on the evaluation set for all of the epochs. It can be observed that the network gains the most performance at the starting epochs and slows down at later epochs.

Configuration		Epoch	Recall		Precision	
Batch Size	Learning Rate		90% IoU	75% IoU	90% IoU	75% IoU
2	1e-04	45	0.7965	0.9322	0.8099	0.9479
2	1e-05	49	0.7643	0.9388	0.7732	0.9498
2	1e-06	45	0.4955	0.8859	0.5008	0.8955
4	**1e-04**	**47**	**0.8007**	**0.9247**	**0.8155**	**0.9419**
4	1e-05	43	0.7568	0.9247	0.7663	0.9363
4	1e-06	49	0.4301	0.8354	0.4758	0.9241

Table 1: RetinaNet Results The table shows the evaluation results for RetinaNet architecture with different settings of the hyperparameters. The best configuration is selected based on the best 90% IoU recall performance and is highlighted in bold.



(a) Recall-90



(b) Precision-90

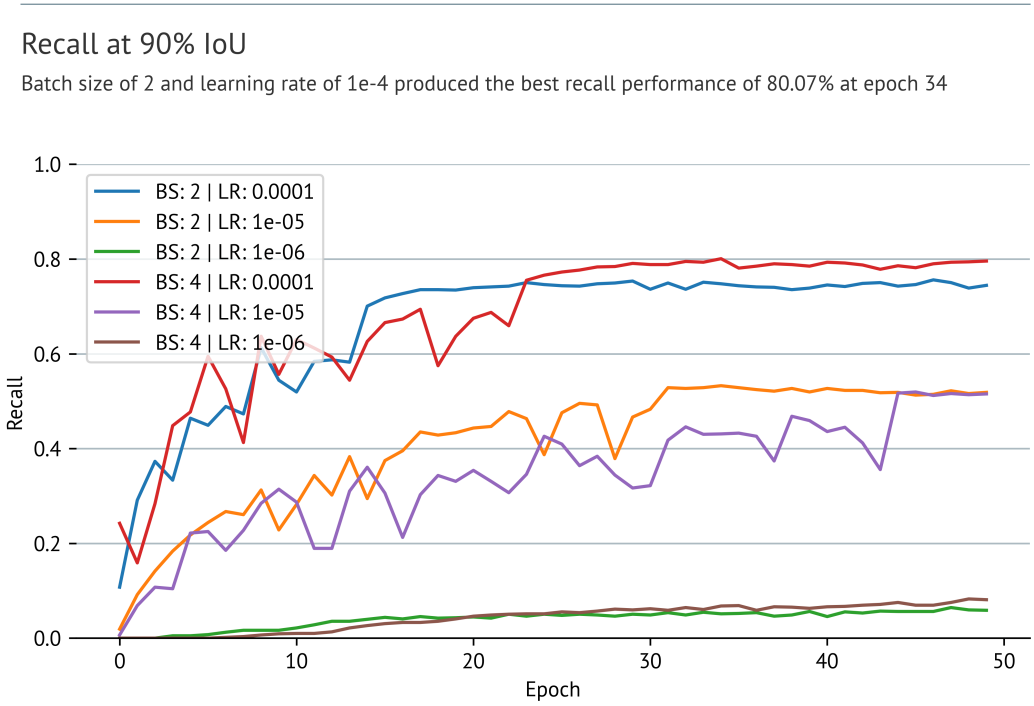
Figure 23: RetinaNet Graphs Precision-90 and Recall-90 performance curves on the evaluation set with respect to the epochs.

RetinaNet v2

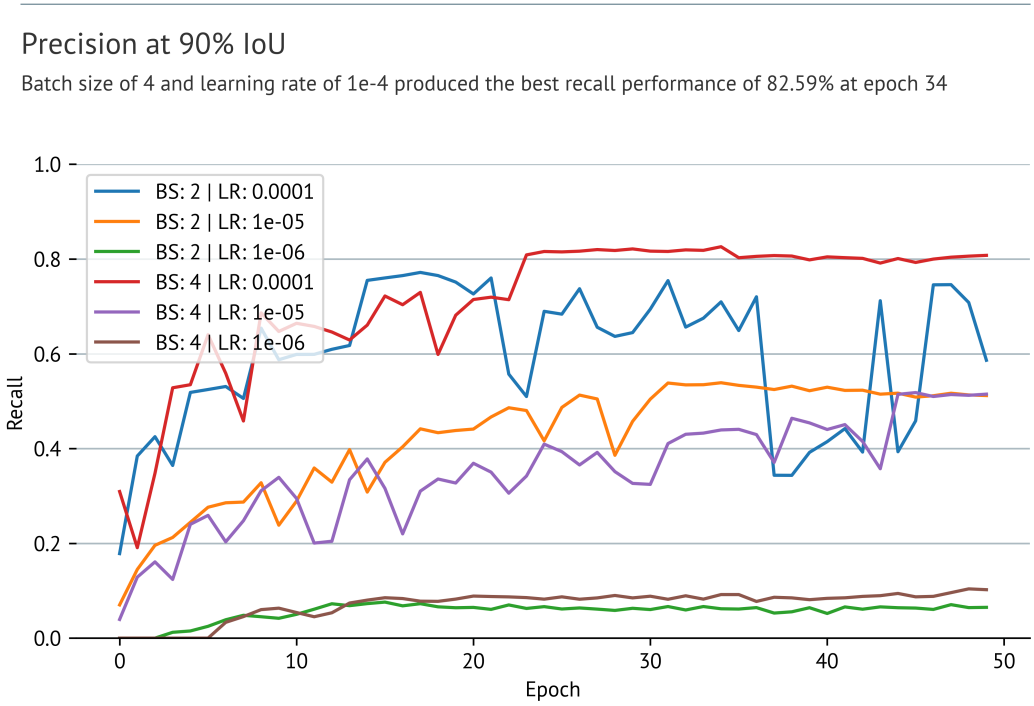
The RetinaNet v2 architecture was an improvement over the RetinaNet architecture while still being a single stage detector. Table 2 shows the results for different hyper-parameters for this architecture. The best model was selected to be with batch size of 4 and learning rate of 1e-04 that produced the highest recall-90 performance of 80% in the 34th epoch. The figure 24 shows the Precision and Recall curves for the evaluation epochs. The curves seem to be divided by the learning rate with 1e-6 proving to be the worst one for this architecture. The learning rate of 1e-4 proved to be the best one but due to a large learning rate there is a possibility that the network will move out of the local maxima that it has found which can be seen with the dip in Precision-90 performance at later epochs (blue curve in Figure 24).

Configuration		Epoch	Recall		Precision	
Batch Size	Learning Rate		90% IoU	75% IoU	90% IoU	75% IoU
2	1e-04	46	0.756	0.9156	0.7455	0.9029
2	1e-05	34	0.5327	0.8519	0.5389	0.8619
2	1e-06	47	0.0645	0.5285	0.0707	0.5788
4	**1e-04**	**34**	**0.8007**	**0.928**	**0.8259**	**0.9573**
4	1e-05	45	0.5194	0.8594	0.5182	0.8573
4	1e-06	48	0.0827	0.5476	0.104	0.6881

Table 2: RetinaNet V2 Results The table shows the evaluation results for RetinaNet v2 architecture with different settings of the hyper-parameters. The best configuration is selected based on the best 90% IoU recall performance and is highlighted in bold.



(a) Recall-90



(b) Precision-90

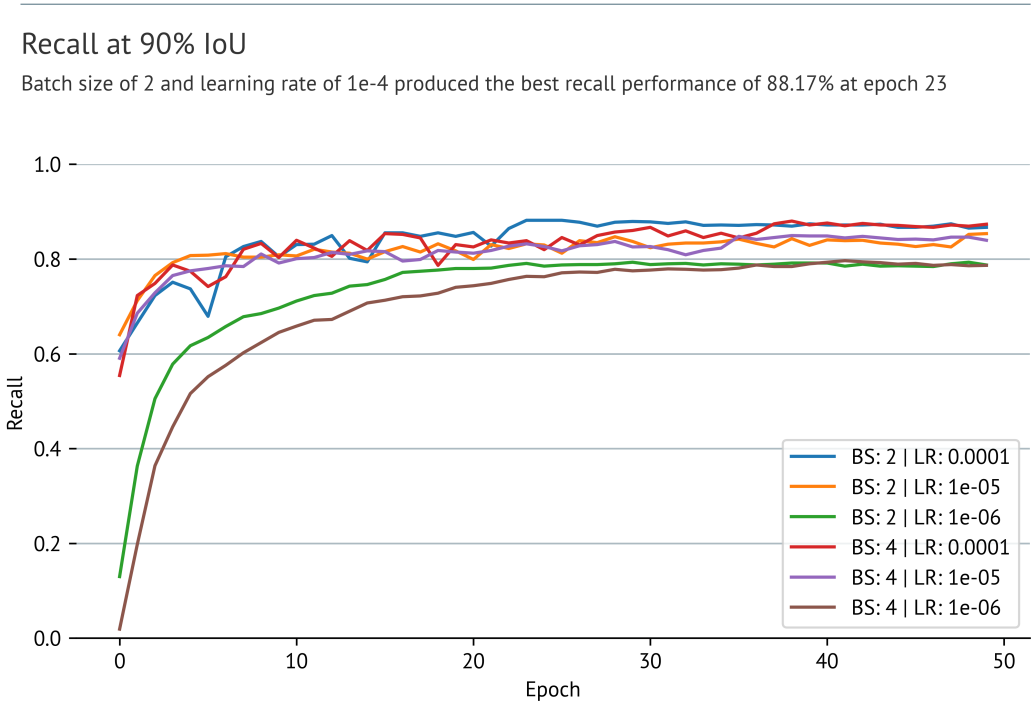
Figure 24: RetinaNet V2 Graphs Precision-90 and Recall-90 performance curves on the evaluation set with respect to the epochs.

FasterRCNN

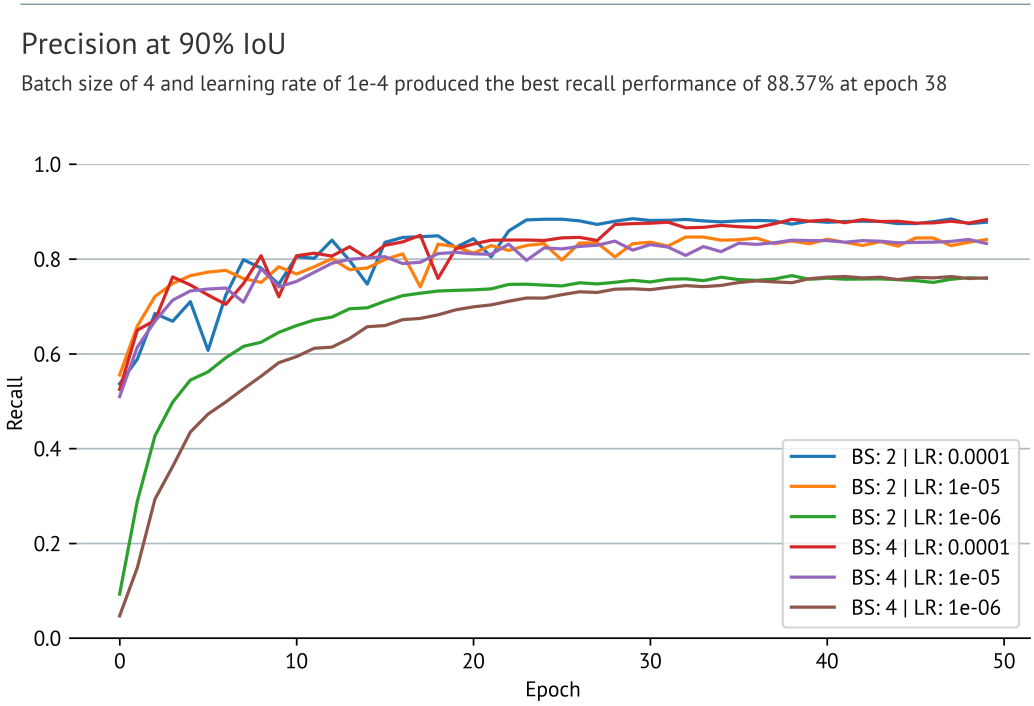
The FasterRCNN was the first of the two two-stage detectors that we considered for the comparison. As it was noted in the background section, the two-stage detectors are slower but are more accurate than single stage detectors and we can observe that in the results from table 3 where across all hyper-parameter combinations the recall-90 and precision-90 performance is better than both RetinaNet and RetinaNet v2 architectures. The best recall-90 performance was 88.17% achieved by the batch size of 2 and the learning rate of 1e-04 at epoch 23. The precision-90 and recall-90 curves for the evaluation procedure are shown in figure 25, once again we observe that 1e-06 learning rate was too low for the networks but the difference is not as big as with RetinaNet and RetinaNet v2 architectures.

Configuration		Epoch	Recall		Precision	
Batch Size	Learning Rate		90% IoU	75% IoU	90% IoU	75% IoU
2	**1e-04**	**23**	**0.8817**	**0.952**	**0.8825**	**0.9528**
2	1e-05	49	0.8536	0.9545	0.8411	0.9405
2	1e-06	29	0.7932	0.957	0.7551	0.911
4	1e-04	38	0.8801	0.9495	0.8837	0.9535
4	1e-05	38	0.8495	0.9529	0.8397	0.9419
4	1e-06	41	0.7965	0.9504	0.7631	0.9105

Table 3: FasterRCNN Results The table shows the evaluation results for Faster-RCNN architecture with different settings of the hyper-parameters. The best configuration is selected based on the best 90% IoU recall performance and is highlighted in bold.



(a) Recall-90



(b) Precision-90

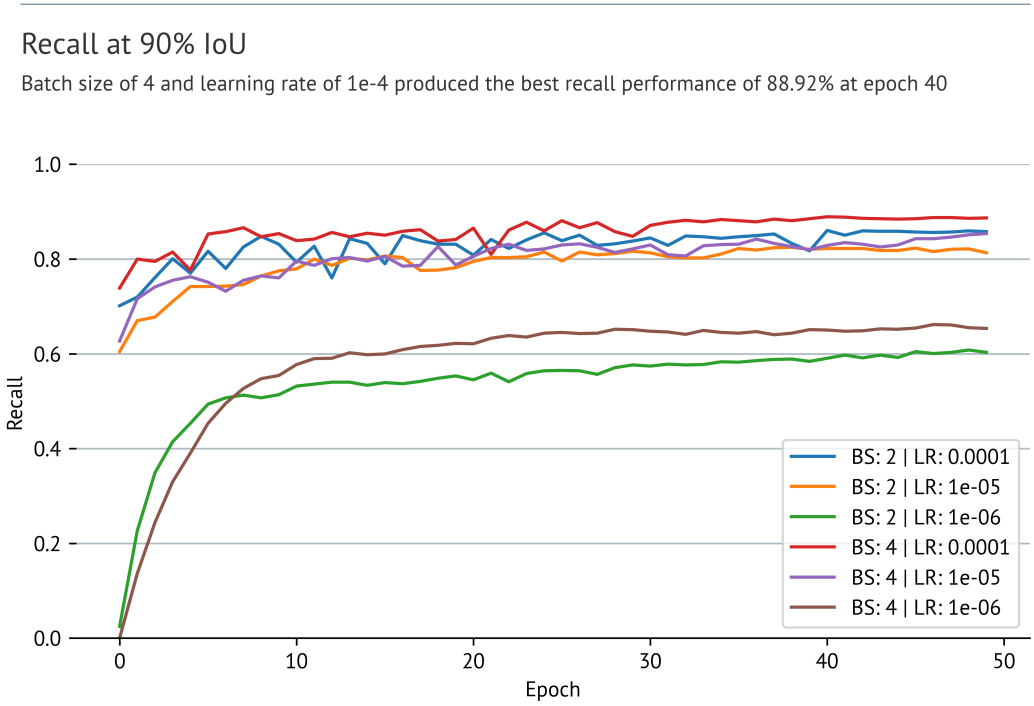
Figure 25: FasterRCNN Graphs Precision-90 and Recall-90 performance curves on the evaluation set with respect to the epochs.

FasterRCNN v2

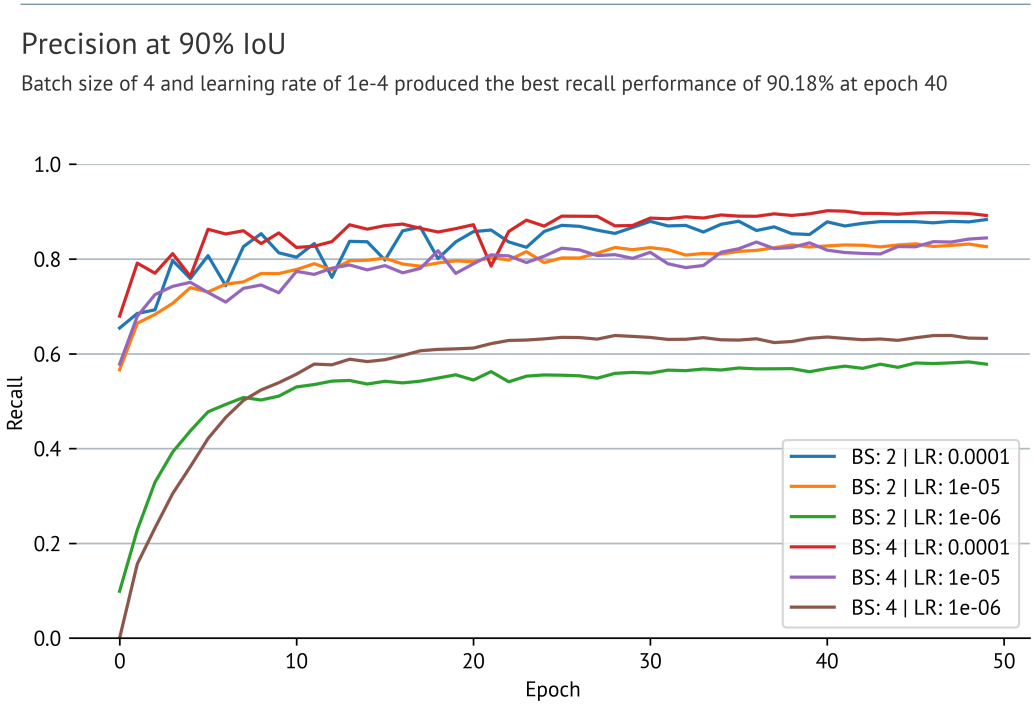
The FasterRCNN v2 architecture was the second of the two-stage objectors that we tested. It gained marginal increase in performance over FasterRCNN. The best recall-90 performance of 88.92% was achieved by a batch size of 4 with learning rate of 1e-04 at epoch 40. The figure 26 shows the recall-90 and precision-90 curves for the different runs. The trend is very similar to FasterRCNN with the learning rate of 1e-06 being noticeably bad for this architecture.

Configuration		Epoch	Recall		Precision	
Batch Size	Learning Rate		90% IoU	75% IoU	90% IoU	75% IoU
2	1e-04	40	0.8602	0.9322	0.8784	0.9519
2	1e-05	38	0.8246	0.9313	0.8295	0.9368
2	1e-06	48	0.6079	0.8759	0.5829	0.8398
4	**1e-04**	**40**	**0.8892**	**0.9512**	**0.9018**	**0.9648**
4	1e-05	49	0.8536	0.9429	0.8445	0.9329
4	1e-06	46	0.6617	0.8941	0.6385	0.8627

Table 4: FasterRCNN V2 Results The table shows the evaluation results for FasterRCNN v2 architecture with different settings of the hyper-parameters. The best configuration is selected based on the best 90% IoU recall performance and is highlighted in bold.



(a) Recall-90



(b) Precision-90

Figure 26: FasterRCNN V2 Graphs Precision-90 and Recall-90 performance curves on the evaluation set with respect to the epochs.

Summary

After the identification of best hyper-parameter setting for each model architecture, the next step is to compare the model architecture themselves. This comparison is performed in table 5. The best model architecture, as can be observed from the table, is FasterRCNN v2 with the recall-90 performance of 88.92%. It is evidently clear from the results that two-stage detectors have performed better than single stage detectors across all the metrics being considered.

Architecture	Configuration		Epoch	Recall		Precision	
	BS	LR		90% IoU	75% IoU	90% IoU	75% IoU
RetinaNet	4	1e-04	47	0.8007	0.9247	0.8155	0.9419
RetinaNet v2	4	1e-04	34	0.8007	0.9280	0.8259	0.9573
FasterRCNN	2	1e-04	23	0.8817	0.9520	0.8825	0.9528
FasterRCNN v2	**4**	**1e-04**	**40**	**0.8892**	**0.9512**	**0.9018**	**0.9648**

Table 5: Model Architecture Comparison This table compares the best runs of the models being compared. 'BS' stands for Batch Size and 'LR' stands for learning rate. The best model architecture is selected based on Recall performance at 90% IoU and is highlighted in bold.

It must be noted here that, the best model based on the best recall-90 performance is selected which is a very strict measure and it requires the predicted bounding boxes to have a very high overlap with ground-truth bounding boxes. In reality, since the ground-truth bounding boxes were also labelled by a human, there can be minor inconsistencies in the table regions or boundaries. In addition, since the way the tables are generated, there is naturally a gap between the values of the tables and their boundaries. This means that even if the 90% IoU overlap criteria is not met by the prediction made by the network, it might still successfully contain all of the values of the tables which can then be extracted. Considering this, the recall performance at 75% IoU overlap is a more realistic metric in real world scenarios. The weights of the best model selected were saved and incorporated into the main pipeline.

5.3 Experiment - Table Classification

The table classification was part of the 'Final Step' discussed in section 4.4. With this experiment, the best word vectoriser and classifier combination was selected for detecting the type of the tables from its raw values.

5.3.1 Evaluation Metrics

Table classification is a multi-class classification problem. To compare the models, the metrics commonly used to evaluate multi-class classification problems are used. The basic definitions and the mathematical formulations for these metrics have already been presented in the previous section. They will be briefly discussed here to put them in context for this task.

- **Accuracy**: The implementation of this metric has been described in section 5.1. For table classification, there are multiple class labels, but accuracy has been calculated over all the class labels collectively in order to provide a broad overview of the performance of the models.

- **Precision**: The implementation of this metric has been described in section 5.1. For this task, the precision is calculated per class label in order to gauge the performance of the model individually on all classes.

- **Recall**: The implementation of this metric has been described in section 5.1. Similar to precision, this metric has been calculated individually for all the class labels.

- **F1-Score**: In order to cover for the deficiencies and the failure modes of the metrics mentioned above, precision and recall, F1-Score has also been calculated

for each class individually. The selection for the best performing model was made based on this metric.

5.3.2 Experiment Setup

Dataset

The dataset used for training and evaluation consisted of 60 randomly selected excel files containing raw extracted values. These 60 excel files contained 215 tables in total. These tables were labelled manually for 4 class labels. More information about these classes are mentioned below.

- **Electrical Characteristics**: The tables belonging to this category contain the electrical characteristics of the solar cell. For example, the open circuit voltages.

- **Thermal Characteristics**: This category indicates the tables that contain the temperature coefficients for the solar cells.

- **Mechanical Characteristics**: The tables belonging to this category contain the mechanical characteristics of the solar cells such as the thickness and the dimensions of the panel.

- **Other**: Any table that does not belong to the three categories above is categorised as 'Other'. An example could be the table that contains intensity dependence values.

The distribution of number examples per class in the dataset is shown in table 6. It can be observed, the dataset is well balanced and should provide enough examples for all classes. The 'Electrical Characteristics' class has the most number of examples as this is the table that is almost always present in the data sheets.

Class	Number of Examples
Electrical Characteristics	62
Thermal Characteristics	36
Mechanical Characteristics	40
Other	77

Table 6: Table Classification Dataset Class Distribution The table shows the distribution of the example tables per class. The top three classes are the tables of the highest interest for the application. All other tables are grouped together in the 'Other' class.

Comparisons

As previously mentioned in section 4.4.1, the two word vectorisation techniques (Count Vectorisation and TF-IDF) and the two classifiers (K-Nearest Neighbours and Naive Bayes) were cross compared to find the best performing combination.

To evaluate the combinations on the full dataset, K-Fold Cross Validation with a 5-way split was used, whereby each combination of classifier and word-vectoriser was trained and tested 5 times on the full dataset, changing the examples that were used for training and testing each time without repeats. The results for the 5 folds were averaged and the best performing combination was selected.

5.3.3 Results

The results from the 5-fold cross validation of the classifier and word-vectoriser combination are presented in this section.

K Nearest Neighbours with Count Vectoriser

Table 7 shows the results for this combination. The F1-Score of all classes is consistent. The precision of first three classes is high but low for the fourth class. The recall

score for all three classes of interest is not very high and this might lead to values being not extracted.

Class	Precision	Recall	F1-Score
Electrical Characteristics	0.92	0.87	0.89
Thermal Characteristics	0.97	0.71	0.81
Mechanical Characteristics	0.95	0.87	0.89
Other	0.75	0.93	0.83
Overall Accuracy			0.86

Table 7: K-Nearest Neighbours with Count Vectoriser Results The table shows the averaged evaluation results of 5 Fold Cross Validation performed on the dataset.

K Nearest Neighbours with TF-IDF

Class	Precision	Recall	F1-Score
Electrical Characteristics	0.95	0.6	0.72
Thermal Characteristics	1.00	0.78	0.87
Mechanical Characteristics	1.00	0.65	0.77
Other	0.60	0.98	0.74
Overall Accuracy			0.77

Table 8: K-Nearest Neighbours with TF-IDF Vectoriser Results The table shows the averaged evaluation results of 5 Fold Cross Validation performed on the dataset.

Table 8 shows the results for this combination. The results follow the same trend as that for the K Nearest Neighbours and Count Vectoriser combination. An even sharper contrast between the precision and recall values can be observed for this combination as compared to the K Nearest Neighbours and Count Vectoriser combination with recall being very poor for all the classes of interest. This combination will not be able detect all of the tables properly. The poor performance is also reflected in accuracy.

Naive Bayes with Count Vectoriser

Table 9 shows the results for this combination. In these results, an increase in model performance as compared to K Nearest Neighbours across all metrics can be observed. The combination is performing notably worse on the 'Thermal Characteristics' class. As this is a class of interest, an increase in F1-Score for it would be beneficial.

Class	Precision	Recall	F1-Score
Electrical Characteristics	0.93	0.96	0.95
Thermal Characteristics	0.78	0.91	0.80
Mechanical Characteristics	0.93	1.0	0.96
Other	0.96	0.80	0.85
Overall Accuracy			0.89

Table 9: Naive Bayes with Count Vectoriser Results The table shows the averaged evaluation results of 5 Fold Cross Validation performed on the dataset.

Class	Precision	Recall	F1-Score
Electrical Characteristics	0.92	0.98	0.95
Thermal Characteristics	1.00	0.91	0.95
Mechanical Characteristics	0.93	0.97	0.94
Other	0.94	0.92	0.93
Overall Accuracy			0.94

Table 10: Naive Bayes with TF-IDF Vectoriser Results The table shows the averaged evaluation results of 5 Fold Cross Validation performed on the dataset.

Naive Bayes with TF-IDF

Table 10 shows the results for this (Naive Bayes with TF-IDF) combination. It shows the best performance across all the metrics compared to all the combinations

above. An increase in performance for the 'Thermal Characteristics' class can also be noted. A mid 90 percent F1-Score for all class values can be observed which is very encouraging.

Summary

After observing the results for F1-score across different classes, the combination of Naive Bayes with TF-IDF was selected as the best model and word-vectoriser combination. The parameters for the model and the word-vectoriser were saved and incorporated into the main pipeline.

5.4 Experiment - Complete Pipeline Evaluation

The solution to the problem of extracting information from PDF documents keeping in view the variability that occurs between different data sheets was divided into further sub-problems or sub-steps as explained in detail in chapter 4. Two critical steps were machine learning based approaches and their experiments were mentioned above individually. Other steps include rule-based instructions for final extraction and validation of the values.

All these steps combine to form the complete end-to-end pipeline that can take in PDF documents and output the values of interest. This section will present the evaluation of the complete pipeline that includes the machine learning models that were trained, the table recognition libraries that were used and the rule-based instructions in the 'Final Step' of the pipeline. This evaluation experiment will also provide the opportunity to compare the approaches of table recognition mentioned in Section 4.3 by fixing other approaches.

5.4.1 Evaluation Metrics

The metrics used for the evaluation of the complete pipeline are familiar as they are the same ones that were used for the case of Table Classification. These metrics are briefly described below in the context of this experiment.

- **Precision**: The basic concept and the mathematical formulation for Precision is the same as that discussed in section 5.1. In the context of this experiment, Precision will stand for the measure that indicates out of all the values that were extracted by the pipeline, how many were actually correct and should have been extracted. The way to cheat this metric would be to be cautious (not extract a lot of values). This will result in a very high Precision score but, of course, not all the values will be extracted.

- **Recall**: Similar to Precision, the basic concept and the mathematical formulation for Recall is the same as that discussed in section 5.1. In the context of this experiment, Recall stands for the measure of how many of the values of interest were actually extracted successfully by the network. The way to cheat on this metric will be to extract everything from the document, even if it does not belong to the property type of interest. That will achieve a very high Recall score but the extracted values will not be very useful.

- **F1-Score**: To cover for the deficiencies and the cheating scenarios of both Precision and Recall, the harmonic mean between them can be calculated, as mentioned in section 5.1, that can better inform us about the overall performance of the complete pipeline.

5.4.2 Experiment Setup

Dataset

For evaluating the complete pipeline, 10 PDF documents were selected. The selection was made to increase diversity in the documents and include different styles of documents. The values of interest were then extracted manually which became the ground-truth and were then saved as a YAML file that are easy to read and manipulate by humans.

Comparisons

The 10 PDF documents mentioned above are presented as input to the pipeline. The files are first converted into images and tables coordinates are detected from these images. Afterwards, the table recognition part is performed by three different approaches i.e. baseline, camelot and tabula. The raw values obtained from the table recognition step are fed into the final step and values of interest are then validated and extracted. In the whole process, everything was kept fixed except the approaches used for table recognition. This allowed a fair comparison of these techniques. The results of this evaluation are presented in the following section.

5.4.3 Results

The results of the evaluation of the complete pipeline with comparison of different approaches are shown in table 11. It can be observed that both Camelot and Tabula perform much better than the baseline approach. The best approach for table recognition, however, is Camelot with F1-Score of 97.31% for the complete end-to-end pipeline.

Since everything but the table recognition approaches were fixed, these results are not only representative of the performance of the complete pipeline but also the table recognition approaches. Thus it is evident from the results that table recognition also plays a critical role in the pipeline and can make or break the results.

Approach	Precision	Recall	F1-Score
Baseline	0.7714	0.7066	0.7376
Camelot	**0.9986**	**0.9490**	**0.9731**
Tabula	0.8405	0.7769	0.8074

Table 11: Complete Pipeline Evaluation Results The tables shows the results from the evaluation of the complete pipeline with different approaches for table recognition used. The best approach is highlighted in bold.

6 Conclusions and Future Work

Deep Learning based Object Detectors are well suited for Table Detection As is evident from the results mentioned in section 5.2, all the object detector models quickly learned to detect the tables in the images with two-stage detectors outperforming single-stage detectors.

Rule-based Table Recognition utilities are good enough for simple tables Two popular rule-based utilities were compared against the baseline for recognising tables in section 5.4. It was identified that this step is very crucial and can make or break the results of the complete pipeline. More sophisticated table recognition utilities Camelot and Tabula performed better than a simple implementation in the form of a baseline.

The final validation and extraction steps need to cover the deficiencies of table recognition utilities The deficiencies of the table recognition utilities were noted as being the bottleneck for correct extraction of values. These deficiencies can manifest themselves in the form of empty cells in the extracted tables or incorrect merging of cells. These deficiencies need to be catered for in the final step.

Text Classification is a suitable technique for determining table type According to the results mentioned in section 5.3, the combination of TF-IDF and Naive Bayes proved to be suitable for classifying tables based on the text contained in them.

In the process of development and evaluation of the pipeline, some areas were identified that can be explored in the future:

- **Training and evaluation of pipeline in other domains**: Currently, the complete pipeline was developed and tested only for solar cell and module data sheets. However, there are other domains that also have similar data sheets, for example solar inverters and batteries. The approach devised here is general and it would be interesting to examine how it performs on data sheets from other domains as well.

- **Improvements in Table Recognition**: The table recognition using deep learning is still an active area of research and, currently, no off-the-shelf tool is available. It would be worthwhile to explore the problem of table recognition from the perspective of deep learning to improve the performance of the pipeline and to make extraction of values from complex tables (those containing merged columns and rows) also viable.

- **Improvements in Table Classification**: Even though, the text classification techniques performed reasonably well for the task of Table Classification, improvements can be made by using the numerical values as additional features.

7 Acknowledgments

I would like to thank Fraunhofer ISE for providing me the opportunity to write this master's thesis. I have had tremendous help and support while working on it. Specifically, I would like to thank:

- Dr.-Ing. Christian Reichel for his constant advice and constructive feedback throughout the course of this thesis.
- Dr. Patrick Brosi for providing me invaluable feedback on my thesis.
- Prof. Dr. Hannah Bast for being the examiner for my thesis.
- Benedictus Dimas Dwitya Artha for his help in the initial phase of the thesis.
- My wife, Maryam, for graciously donating her time to correct my incoherent sentences and misspelled words and her constant emotional support.
- My parents for their unconditional emotional support.

Bibliography

[1] B. D. D. Artha, "Data mining im bereich von photovoltaik-technologie," 2021.

[2] J. H. Shamilian, H. S. Baird, and T. L. Wood, "A retargetable table reader," in *Proceedings of the fourth international conference on document analysis and recognition*, vol. 1, pp. 158–163, IEEE, 1997.

[3] B. Klein, S. Gokkus, T. Kieninger, and A. Dengel, "Three approaches to" industrial" table spotting," in *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pp. 513–517, IEEE, 2001.

[4] T. Hassan and R. Baumgartner, "Table recognition and understanding from pdf files," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2, pp. 1143–1147, IEEE, 2007.

[5] Y. Wang, I. T. Phillips, and R. M. Haralick, "Table structure understanding and its performance evaluation," *Pattern recognition*, vol. 37, no. 7, pp. 1479–1497, 2004.

[6] 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, IEEE, 2016.

[7] A. Gilani, S. R. Qasim, I. Malik, and F. Shafait, "Table detection using deep learning," in *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, vol. 1, pp. 771–776, IEEE, 2017.

[8] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed, "Deepdesrt: Deep learning for detection and structure recognition of tables in document images," in *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, vol. 1, pp. 1162–1167, IEEE, 2017.

[9] A. Nurminen, "Algorithmic extraction of data in tables in pdf documents," Master's thesis, 2013.

[10] A. Zucker, Y. Belkada, H. Vu, and V. N. Nguyen, "Clusti: Clustering method for table structure recognition in scanned images," *Mobile Networks and Applications*, vol. 26, no. 4, pp. 1765–1776, 2021.

[11] S. Raja, A. Mondal, and C. Jawahar, "Visual understanding of complex table structures from document images," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2299–2308, 2022.

[12] S. R. Qasim, H. Mahmood, and F. Shafait, "Rethinking table recognition using graph neural networks," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 142–147, IEEE, 2019.

[13] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: a statistical framework," *International journal of machine learning and cybernetics*, vol. 1, pp. 43–52, 2010.

[14] W. B. Cavnar, J. M. Trenkle, *et al.*, "N-gram-based text categorization," in *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, vol. 161175, Las Vegas, NV, 1994.

[15] L. Havrlant and V. Kreinovich, "A simple probabilistic explanation of term frequency-inverse document frequency (tf-idf) heuristic (and variations motivated by this explanation)," *International Journal of General Systems*, vol. 46, no. 1, pp. 27–36, 2017.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[17] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[18] M. E. Maron, "Automatic indexing: an experimental inquiry," *Journal of the ACM (JACM)*, vol. 8, no. 3, pp. 404–417, 1961.

[19] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[20] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings*, pp. 137–142, Springer, 2005.

[21] Y. Kim, "Convolutional neural networks for sentence classification," 2014.

[22] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Comput. Surv.*, vol. 51, sep 2018.

[23] Adobe, "What is PDF?." `https://www.adobe.com/acrobat/about-adobe-pdf.html`, accessed on 2023-02-08.

[24] Wikipedia, "PDF." `https://en.wikipedia.org/wiki/PDF`, accessed on 2023-02-08.

[25] PyMuPDF, "Introduction to PyMuPDF." https://pymupdf.readthedocs.io/en/latest/, accessed on 2023-02-08.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[27] Wikipedia, "tf-idf." https://en.wikipedia.org/wiki/Tf%E2%80%93idf, accessed on 2023-02-19.

[28] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*, vol. 39. Cambridge University Press Cambridge, 2008.

[29] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943.

[30] IBM, "What is a Neural Network." https://www.ibm.com/topics/neural-networks, accessed on 2023-02-14.

[31] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[32] S. Knowledge, "Perceptrons – These Artificial Neurons are the Fundamentals of Neural Networks." https://starship-knowledge.com/neural-networks-perceptrons, accessed on 2023-02-14.

[33] S. Maheshkar, "What Is Cross Entropy Loss? A Tutorial With Code." https://wandb.ai/sauravmaheshkar/cross-entropy/reports/What-Is-Cross-Entropy-Loss-A-Tutorial-With-Code--VmlldzoxMDA5NTMx, accessed on 2023-02-16.

[34] Wikipedia, "Chain Rule." https://en.wikipedia.org/wiki/Chain_rule, accessed on 2023-02-17.

[35] G. W. Lindsay, "Convolutional neural networks as a model of the visual system: Past, present, and future," *Journal of cognitive neuroscience*, vol. 33, no. 10, pp. 2017–2031, 2021.

[36] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, p. 106, 1962.

[37] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[38] Aphex34, "Typical CNN." `https://commons.wikimedia.org/wiki/File:Typical_cnn.png`, accessed on 2023-02-17.

[39] J. Loy, "Building a simple CNN." `https://www.oreilly.com/library/view/neural-network-projects/9781789138900/8e87ad66-6de3-4275-81a4-62b54436bf16.xhtml`, accessed on 2023-03-27.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[41] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[42] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, pp. 154–171, 2013.

[43] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[44] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[45] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

[46] 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.

[47] Y. Li, S. Xie, X. Chen, P. Dollar, K. He, and R. Girshick, "Benchmarking detection transfer learning with vision transformers," *arXiv preprint arXiv:2111.11429*, 2021.

[48] A. Bindal, "Normalization Techniques in Deep Neural Networks." `https://medium.com/techspace-usict/normalization-techniques-in-deep-neural-networks-9121bf100d8`, accessed on 2023-02-19.

[49] W. Wong, "What is Group Normalization?." `https://towardsdatascience.com/what-is-group-normalization-45fe27307be7`, accessed on 2023-02-19.

[50] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 658–666, 2019.

[51] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li, "Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9759–9768, 2020.

[52] T. Documentation, "tabula-extractor." https://www.rubydoc.info/gems/tabula-extractor/0.7.6, accessed on 2023-04-01.

[53] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755, Springer, 2014.