Master's Thesis

Recognition of Complex Table Structures and Extraction of Tabular Data From Photovoltaic Module Datasheets

Swathi Thiruvengadam

First Examiner: Prof. Dr. Hannah Bast

Second Examiner: Prof. Dr. Holger Neuhaus

Advisers: Dr. Ing. Christian Reichel

Dr. Patrick Brosi

University of Freiburg

Faculty of Engineering

Department of Computer Science

Chair for Algorithms and Data Structures

and

Fraunhofer-Institut für Solare Energiesysteme

Module Technology Department

March 10th, 2025

Writing Period

18. 12. 2024 – 10. 03. 2025

Examiner

Prof. Dr. Hannah Bast

Second Examiner

Prof. Dr. Holger Neuhaus

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.

Freiburg, 10.03.2025

Place, Date

Signature

Abstract

The growing demand for solar energy in recent times calls for the development of efficient approaches to analyze photovoltaic (PV) module datasheets. Critical technical specifications are documented in these datasheets, which often contain unstructured data with inconsistent formatting that makes automated data extraction challenging. Tables with complex structures containing merged cells and multiple headers result in missing or incorrect data extraction if processed through rule-based systems. This thesis addressed these challenges by proposing a pipeline for accurate recognition of complex table structures and extraction of relevant key data.

The proposed pipeline integrates Deep Learning (DL) models capable of Table Detection (TD) and Table Structure Recognition (TSR) with optical character recognition (OCR) to transform PDF datasheets into structured outputs. A transformer-based Detection Transformer (DETR) model was initially trained to localize tables present in documents. This was followed by a second DETR model that was trained to detect rows, columns, merged cells, multi-row/multi-column headers, and other complex structures. An improved canonicalization algorithm was also implemented to process horizontal and dual-axis tables. Post-processing techniques like Naive Bayes classifier and regular expressions pattern matching enabled validation and extraction of relevant data from the datasheets to aid PV module research.

Experiments performed on a custom PV module test dataset show that this pipeline significantly outperforms traditional rule-based extraction methods when processing tables with complex structures. The TD model attained an Average Precision (AP50) of up to 89.5%, and the TSR model attained an AP50 of 71.3%. Furthermore, the TSR model obtained a GriTS_Loc (Location Grid Table Similarity) score of 92.18% on complex tables whcih indicates strong performance in locating complex structures such as merged cells and headers. The experiments also showed the pipeline's robustness to noisy images and layout variations and noted that modifications to the OCR engine and enhancing the canonicalization algorithm could further improve the extraction results. This thesis provides an end-to-end pipeline for accurate and automated large-scale extraction of PV data to support broader sustainable energy initiatives.

Table of Contents

List of Figures

List of Tables

1 Introduction

In today's digital information era, data is a vital resource that drives innovation by optimizing and streamlining processes to enable well-informed decisions. Despite the recent technological advancements in data processing and the significant increase in storage capabilities, the vast amounts of data generated daily are still being stored in unstructured data formats. This has created a need for efficient methods to organize, analyze, and extract meaningful insights from such data. Data mining plays an important role in extracting valuable information from a large pool of readily available data. However, since this data is typically unstructured and formatted for easy human consumption, it lacks standardization, making it difficult for automated machine processing and data extraction.

Photovoltaic module datasheets, which provide important technical specifications such as electrical, mechanical, and thermal characteristics of the PV module in tabular format, are examples of unstructured data. The complexity of tables observed in these datasheets further increases the difficulty of processing this data. Hence, Table Structure Recognition and extraction of tabular data from PV module datasheets is crucial in leveraging this vast amount of technical information to aid the development of efficient PV modules. This thesis focuses on developing a robust pipeline for accurately identifying complex table structures and extracting relevant data from these documents to bridge this gap.

1.1 Motivation

The main motivation behind this thesis stems from the rapidly growing importance of solar energy in recent years to solve the global energy crisis. With the increased demand for renewable energy sources to mitigate the use of fossil fuels and the need to reduce climate change, solar power has become crucial for the transition towards sustainable energy systems. The manufacturing and maintenance cost of PV modules is relatively low compared to other renewable energy sources, and it is easy to scale. Thus, making it versatile for both small residential setups and large-scale industrial installations. Figure 1 illustrates a graph depicting the annual percentages of electricity demand met by PV modules between 2010-2021 and the projected electricity demands for 2050 across regions. This shows a rapid increase in globally installed photovoltaic capacity, which

exceeded 1 terawatt in 2022 [1]. Despite this, it only contributed to 4-5% of the annual global electricity generation, thereby indicating room for further development.

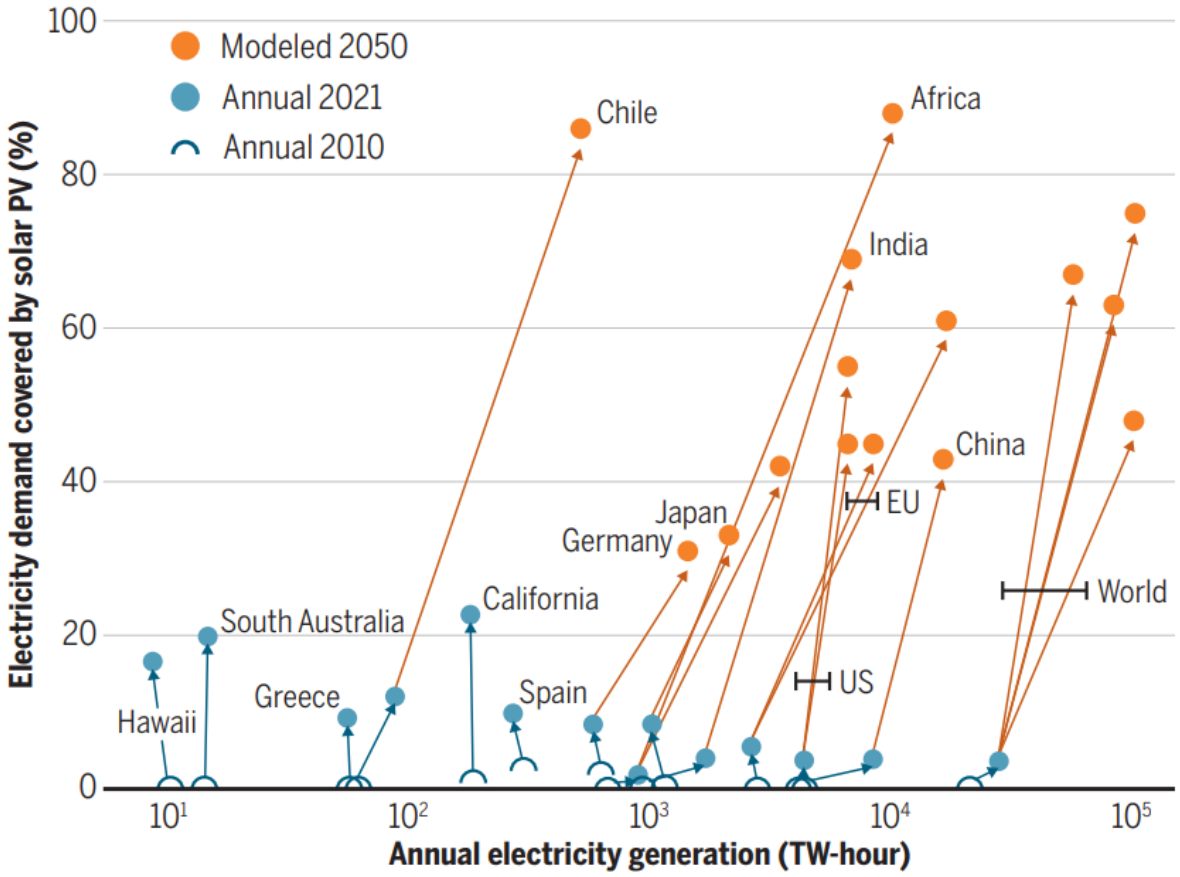


Figure 1: Global growth of Photovoltaic capacity and its contribution to electricity demands [1].

Figure 2 presents future PV capacity projections requiring PV installations to reach 75 TW by 2050 to meet energy needs and climate goals [1]. To achieve this, an annual growth rate of 25% is required over the next decade. This shows the urgent need for efficient methods to process and analyze vast amounts of readily available data and gain useful insights. To support this rapid expansion of the solar sector, it is essential to automate and streamline the analysis of PV module datasheets to drive advancements in PV technology, optimize module designs, and uncover regional performance trends. PV module datasheets hold critical information, but they are rarely standardized and often include complex table layouts, scattered across scanned PDFs or image-based documents.

Enabling quick, accurate extraction and analysis of these datasheets would benefit multiple stakeholders. Access to a comprehensive and structured PV module database enables researchers to conduct in-depth analysis of solar technology advancements, visualize trends, identify under-researched areas, and comparative performance studies. For PV manufacturers and market analysts, this aggregated data would facilitate strategic decision-making, help identify market

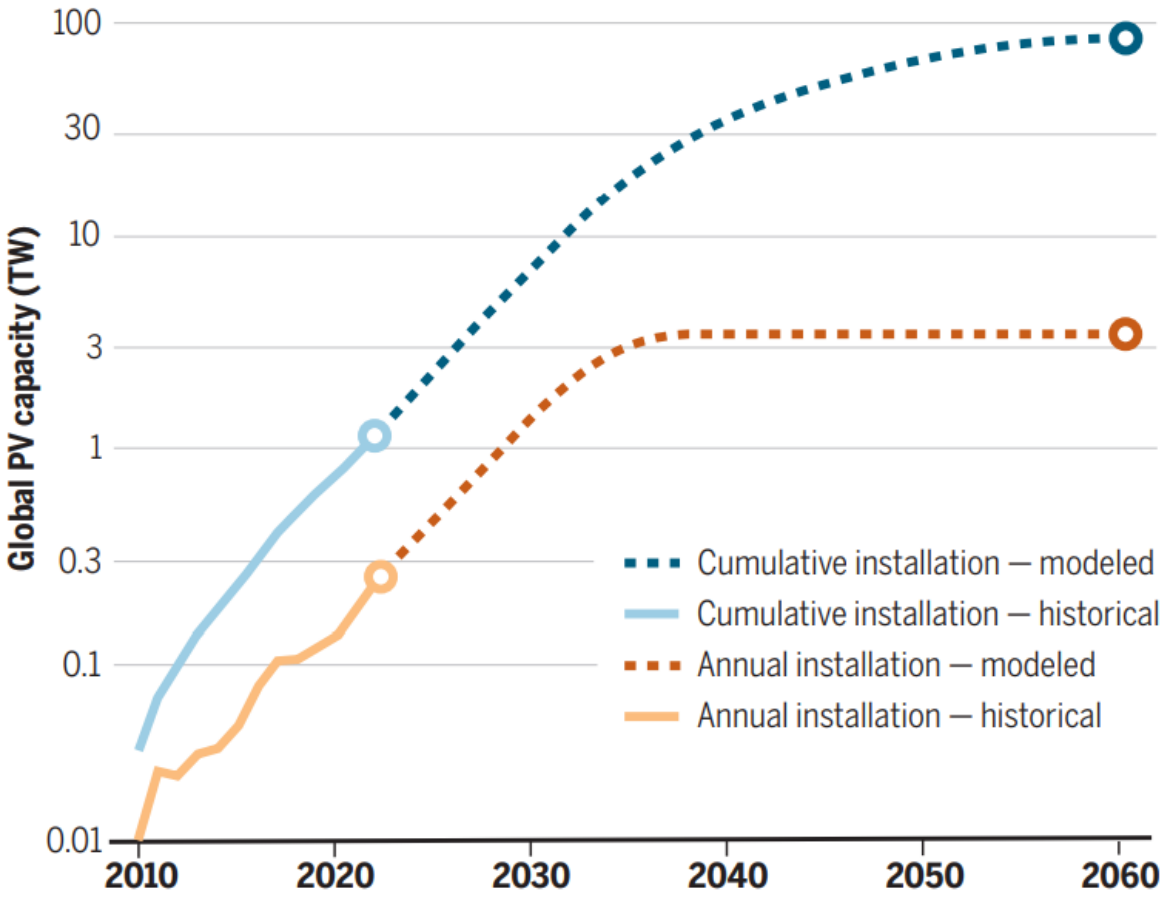


Figure 2: Projected global growth of Photovoltaic capacity [1].

opportunities, and optimize product performance. Policymakers and environmental planners could use these insights to promote solar adoption and achieve energy targets. This thesis aims to contribute to the global effort of expanding renewable energy infrastructure by addressing the critical challenges in data extraction and analysis posed by rule-based systems.

1.2 Problem Statement

Tabular information is an important format of data representation in many fields, including the PV sector due to its easy and concise data representation for human understanding. However, gathering this data from technical documents is often complicated by inconsistent layouts, merged cells, and other structural variations, which can obscure important details [27]. These issues become especially problematic when documents are only available as scanned images or non-searchable image-based PDFs because manual extraction then takes significant time and resources, driving up the risk of errors and oversights [28]. PDF files often lack consistent encoding for

tables, as they are predominantly organized visually rather than tagged for machine reading, resulting in the need for a model to interpret visual layout as a logical structure.

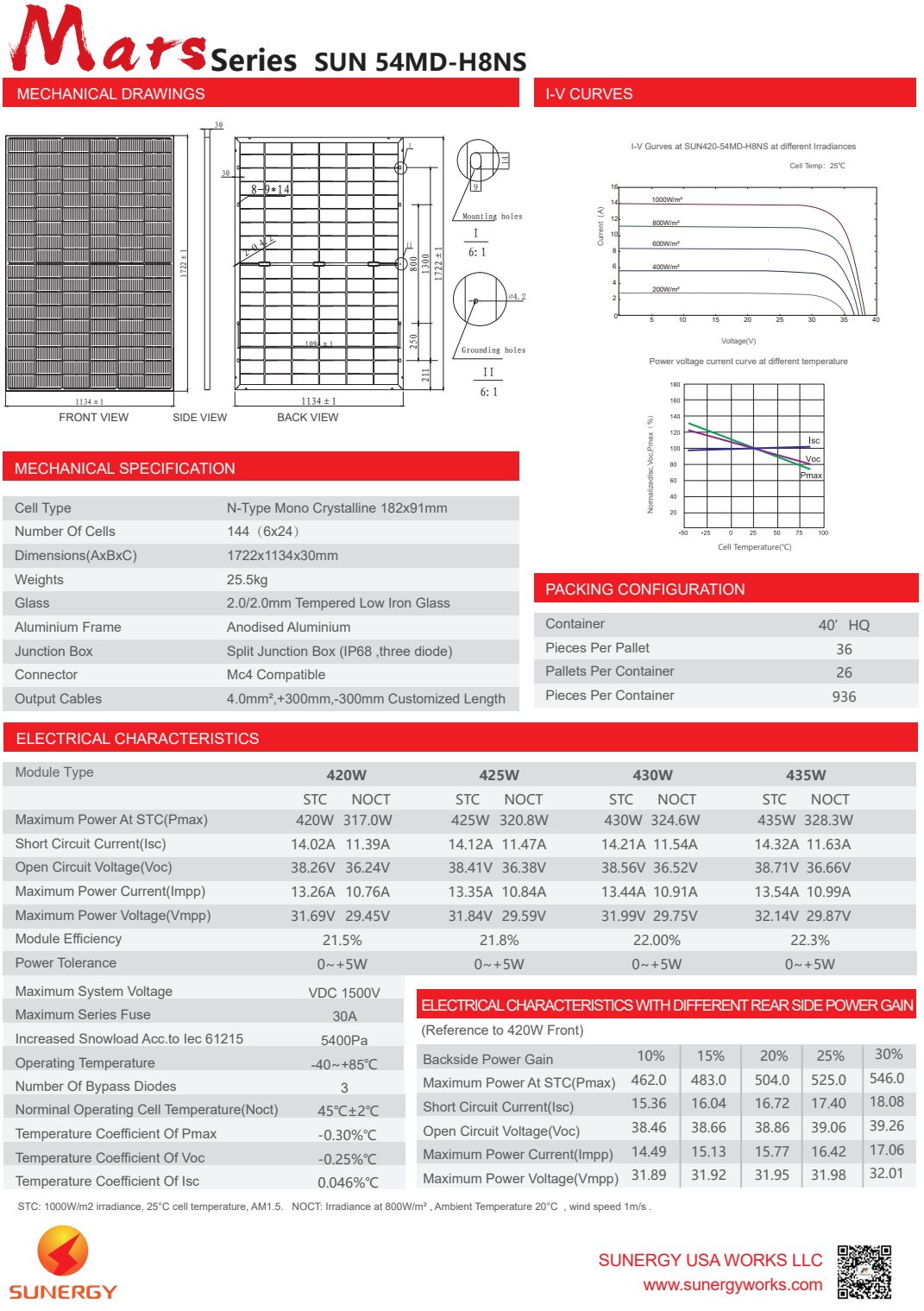


Figure 3: Image-based PV module datasheet with complex table layouts [2].

Lynx

BIFACIAL N-TYPE MONO CRYSTALLINE HALF CUT MODULE – DOUBLE GLASS

RCM-xxx-7DBNG (xxx=410-440)

Electrical Characteristics

POWER CLASS [1]			410		415		420		425		430		435		440	
Testing Condition			STC [2]	*NMOT [3]*	STC	*NMOT*	STC	*NMOT*	STC	*NMOT*	STC	*NMOT*	STC	*NMOT*	STC	*NMOT*
Maximum Power	Pmax	[Wp]	410	*308*	415	*312*	420	*316*	425	*320*	430	*323*	435	*327*	440	*331*
Maximum Power Voltage	Vmp	[V]	31,13	*29,06*	31,32	*29,20*	31,51	*29,34*	31,70	*29,50*	31,88	*29,63*	32,07	*29,79*	32,26	*29,95*
Maximum Power Current	Imp	[A]	13,17	*10,61*	13,25	*10,69*	13,33	*10,76*	13,41	*10,83*	13,49	*10,91*	13,57	*10,98*	13,64	*11,05*
Open Circuit Voltage	Voc	[V]	37,73	*35,84*	37,92	*36,02*	38,11	*36,20*	38,30	*36,38*	38,49	*36,56*	38,67	*36,74*	38,85	*36,93*
Short Circuit Current	Isc	[A]	13,94	*11,23*	13,99	*11,30*	14,07	*11,36*	14,15	*11,42*	14,23	*11,49*	14,31	*11,55*	14,38	*11,62*
Module Efficiency	Eff	[%]	21,00		21,25		21,51		21,77		22,02		22,28		22,54	
Maximum Series Fuse	IR	[A]	30													
Maximum System Voltage	VSYS	[V]	1500V DC (IEC)													

(1) Measurement Tolerances: Pmax (± 3%), Isc & Voc (± 3%) - Power Classification 0/+5W
(2) STC (Standard Testing Condition): Irrandiance 1000W/m², Cell Temperature 25°C, AM 1.5
(3) NMOT (Nominal Operating Module Temperature): Irrandiance 800W/m², NMOT, Ambient Temperature 20°C, AM 1.5, Wind Speed 1m/s

Bi Facial Output (4)

POWER CLASS			410		415		420		425		430		435		440	
			Pmax [Wp]	*Eff [%]*	Pmax [Wp]	*Eff [%]*	Pmax [Wp]	*Eff [%]*	Pmax [Wp]	*Eff [%]*	Pmax [Wp]	*Eff [%]*	Pmax [Wp]	*Eff [%]*	Pmax [Wp]	*Eff [%]*
Power with Backside Gain	+5	[%]	430,5	*22,0%*	435,8	*22,3%*	441,0	*22,6%*	446,3	*22,9%*	451,5	*23,1%*	456,8	*23,4%*	462,0	*23,7%*
	+10	[%]	451,0	*23,1%*	456,5	*23,4%*	462,0	*23,7%*	467,5	*23,9%*	473,0	*24,2%*	478,5	*24,5%*	484,0	*24,8%*
	+15	[%]	471,5	*24,1%*	477,3	*24,4%*	483,0	*24,7%*	488,8	*25,0%*	494,5	*25,3%*	500,3	*25,6%*	506,0	*25,9%*
	+20	[%]	492,0	*25,2%*	498,0	*25,5%*	504,0	*25,8%*	510,0	*26,1%*	516,0	*26,4%*	522,0	*26,7%*	528,0	*27,0%*
	+25	[%]	512,5	*26,2%*	518,8	*26,6%*	525,0	*26,9%*	531,3	*27,2%*	537,5	*27,5%*	543,8	*27,8%*	550,0	*28,2%*
	+30	[%]	533,0	*27,3%*	539,5	*27,6%*	546,0	*28,0%*	552,5	*28,3%*	559,0	*28,6%*	565,5	*29,0%*	572,0	*29,3%*

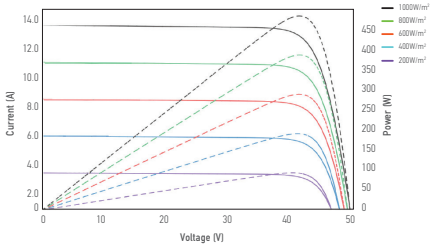
(4) Bifaciality Factor > 80% - Back-side power gain depends upon the specific project albedo - Efficiency is according to the surface of the module

Mechanical Data

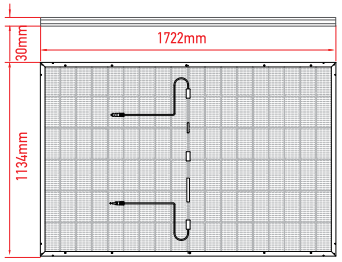
Dimensions	1722 mm x 1134 mm x 30 mm
Weight	24,7 Kg
Cell Type	N-type - 182mm x 91mm (2 x 54 Pcs) - M10
Front Glass	2.0 mm Tempered and low iron glass + ARC
Rear Side	2.0 mm Tempered and low iron glass
Frame	Anodized Aluminium Alloy
Junction Box	IP68, 3 Bypass diodes
Connector	MC4 compatible
Output cable	4mm² - Length: = 350mm or can be customized

I-V Curve

The module relative power loss at low light irradiance of 200W/m² is less than 3%.



Dimensions



Temperature Characteristics

Pmax Temperature Coefficient	-0.290% / °C
Voc Temperature Coefficient	-0.250% / °C
Isc Temperature Coefficient	+0.045% / °C
Operating Temperature	-40~+85 °C
Nominal Operating Module Temperature (NMOT)	42 ± 2 °C

Packing Configuration

Container	40'HC
Pieces per Pallet	37
Pallets per Container	26
Pieces per Container	(37+37)x13=962 pcs

www.recom-tech.com

**Release RCM-xxx-7DBNG(xxx=410-440)-1L-M10-30-5G-15H-017-2023-03-v4.0

Figure 4: Image-based PV module datasheet with high volume of data [3].

The sheer volume of data present in the image-based PDF datasheets along with the inconsistent table layout design, as observed in Figure 3 and Figure 4 respectively, make manual data extraction challenging. Furthermore, table boundary lines between rows and columns may be absent, and tables may span multiple pages, further complicating accurate data capture [28]. Although feasible on fewer datasheets, manual data extraction can be time consuming, labor intensive, and highly prone to human error with increasing dataset size.

Another common problem is that different documents present tables in different styles, including variations in fonts, colors, and borders, which can carry additional meaning as seen in Figure 3 and Figure 4 [29]. However, this additional meaning is not always captured in rule-based extraction methods like Tabula, developed by Aristarán et al. [30], or Camelot, developed by Atlan Labs [15].

These problems become especially pressing in the photovoltaic sector, where ongoing growth in solar energy production depends on accurate and up-to-date information for decision making. A single error in data extraction can lead to misinformed decisions in data-critical tasks such as module design and optimization, influence procurement choices, or cause delays in policy planning [27]. Consequently, the core problem is the absence of a reliable, automated system to accurately recognize and extract data from the wide variety of tabular layouts found in PV module datasheets.

1.3 Thesis Objectives

Based on the motivation and problem statement discussed above, this thesis aims to develop an end-to-end pipeline capable of extracting tabular data from documents with varying styles and complex table layouts. This work pursues the following objectives:

- Developing an automated Deep Learning system capable of locating tables in densely-packed PV module datasheets, including those that span multiple pages to extract relevant data. This includes developing strategies to parse the individual detected table to recognize the table's internal components, tackle multiple headers, handle merged cells and missing data, and manage inconsistent header alignments, thus improving extraction reliability.
- Proposing a pipeline to clean, validate, and transform the extracted data into a standardized format to enable further analysis and comparison across different datasheets.

By addressing these objectives, this thesis seeks to establish a foundational framework for automated data extraction, particularly in the field of solar energy research and generally in other domains with complex document structures and styles.

1.4 Thesis Structure

This thesis is systematically organized to present the development and evaluation of an automated tabular extraction pipeline from PV module datasheets. The structure of this thesis is as follows:

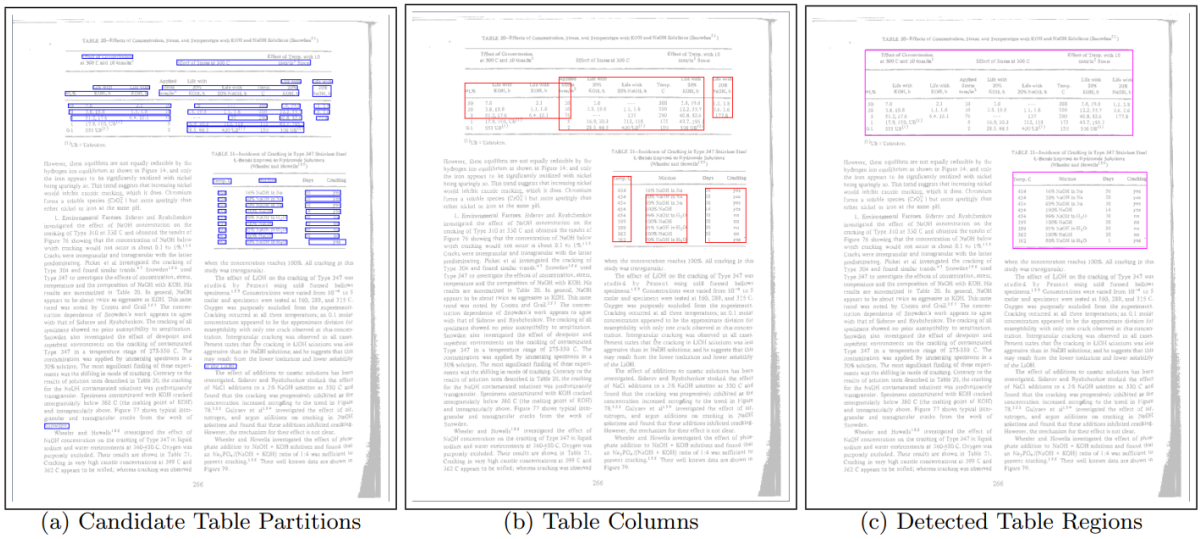
- Chapter 2 presents the relevant literature and recent advancements that serve as the foundation for this thesis.
- Chapter 3 provides the essential theoretical background for understanding the fundamental concepts explored in this thesis, including the models utilized.
- Chapter 4 describes the complete end-to-end inference pipeline implementation and methodology.
- Chapter 5 outlines the experimental setup and evaluation metrics used to assess model performance on various tasks.
- Chapter 6 compares this pipeline with existing methods, analyzes the strengths of the proposed approach, discusses its limitations, and suggests future improvements.
- Chapter 7 concludes this thesis by summarizing key contributions and findings.

2 Related Work

This section reviews existing research and technological advancements in Table Detection (TD), Table Structure Recognition (TSR), and the existing photovoltaic (PV) data extraction pipeline created using Lightning-Table, introduced by Malik [13]. It also highlights the evolution of approaches from rule-based systems to Deep Learning and transformer-based methods, which have significantly improved the accuracy of recognizing and extracting data from complex tables.

2.1 Table detection

Table detection is the process of locating regions within documents that contain tables. It is a crucial first step in extracting tabular and structured data from documents. However, accurately identifying table boundaries within documents can be challenging due to the lack of explicit markers and tags defining a table structure. The varying table layouts and document design can further complicate this step.



(a) Candidate Table Partitions (b) Table Columns (c) Detected Table Regions

Figure 5: Result of various steps in Tesseract's layout analysis algorithm for table detection proposed by Shafait et al. [4].

Initially, rule-based systems that analyzed document layouts and text patterns were used by early researchers for identifying tables in documents. Chandran and Kasturi [31] proposed one such approach which involved extracting the horizontal and vertical lines along with white streams as substitutes for any missing demarcation lines to detect tables. Similarly, Shafait et al. [4] detected

tables with varying layouts in scanned document images by analyzing document layouts using Tesseract, which was originally developed at Hewlett-Packard [32]. Figure 5 depicts the results of the different steps of the table detection algorithm proposed by Shafait et al. [4]. It demonstrates a high accuracy in detecting document elements including tables but a major limitation of this approach was that it was a traditional technique and not data-driven.

Harit and Bansal [5] proposed yet another rule-based technique for table detection based on the identification of unique table start and trailer patterns. Figure 6 shows the flow diagram of the proposed method. However, this approach did not work as intended when the table start patterns were not unique in the document images.

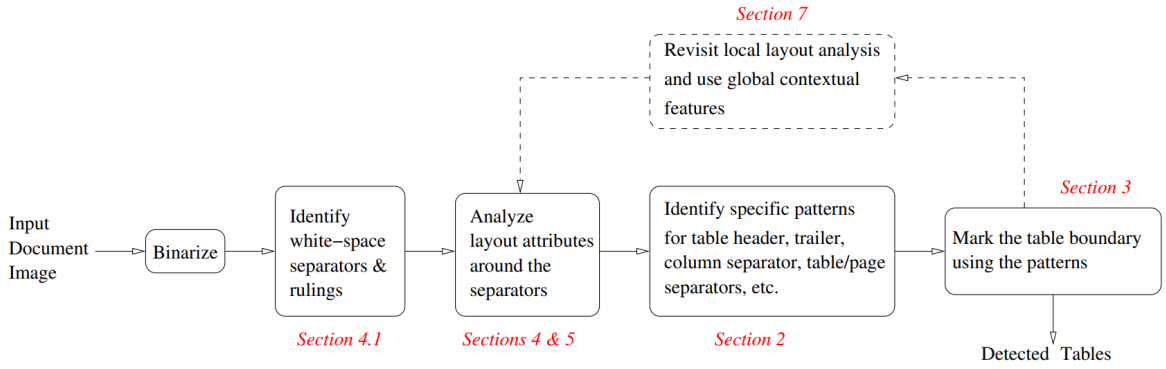


Figure 6: Flow diagram of the rule-based system proposed by Harit and Bansal [5].

While the rule-based approaches worked well on structured tables, it often struggled with real-world documents featuring complex table layouts that lack consistent row or column demarcations, which were common in technical datasheets. This led to the emergence of Machine Learning based approaches to improve detection accuracy. Rashid et al. [33] developed a pipeline using a pre-trained neural network model to classify tables. The output was further enhanced by applying contextual post-processing on each element to correct the classification errors, and this approach achieved an accuracy of 97% on the test dataset.

With the success of DL techniques in computer vision, convolutional neural networks (CNNs) have become a standard for table detection. Schreiber et al. [34] presented an approach that used region-based CNNs (R-CNN) to localize and classify tables within document images by applying transfer learning and domain adaptation to existing object detectors to enhance the model's robustness across different document layouts.

Following this, Gilani et al. [6] introduced a faster R-CNN based approach, where the document image was first transformed and then fed into a fine-tuned CNN model. The feature map outputs were fed into the Region Proposal Network (RPN) for proposing candidate table regions. These

regions along with the convolutional feature map were passed to a fully connected detection network to identify tables as seen in Figure 7. This approach enhanced detection speed and resulted in high precision on document images with varying layouts. Their emphasis on generalizing across various document styles makes it highly relevant to this thesis, where table layouts vary significantly across manufacturers. Similarly, Ma et al. [7] proposed using CornerNet as a new region proposal network to generate table proposals for Faster R-CNN, to improve the localization accuracy. Figure 8 depicts the overall architecture of the CornerNet-FRCN based table detection approach.

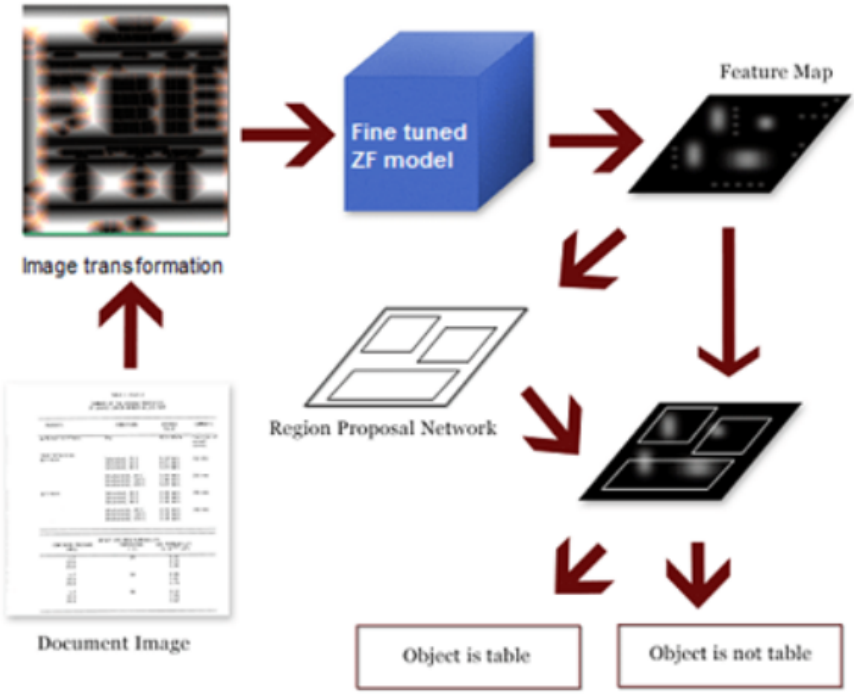


Figure 7: Table detection approach utilized by Gilani et al. [6].

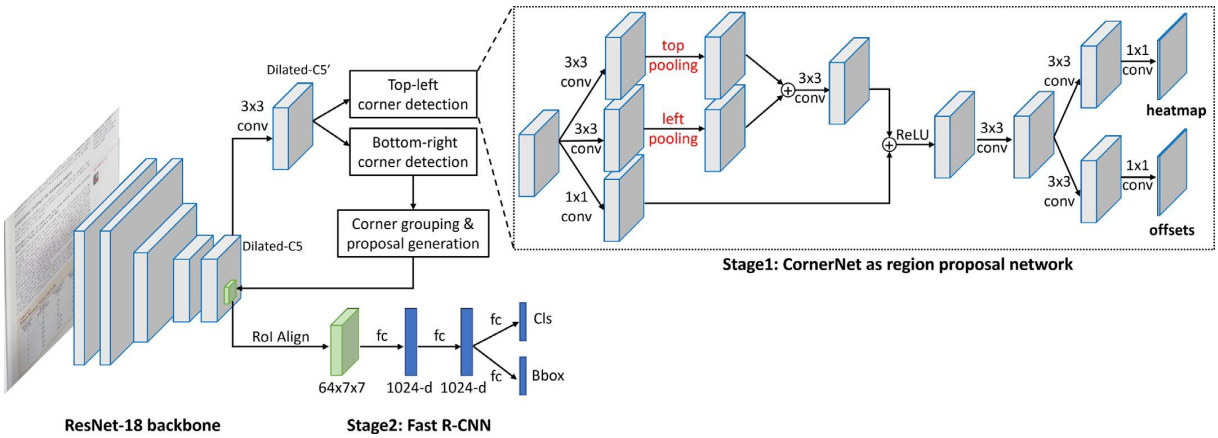


Figure 8: CornerNet-FRCN architecture proposed by Ma et al. [7].

Lately, the introduction of transformer-based models has addressed the limitations of CNN-based methods, particularly in handling complex table structures, and improved the contextual understanding of table layout. Paliwal et al. [8] proposed TableNet, an end-to-end image semantic segmentation model to jointly learn the structure and layout of tables. This architecture, as seen in Figure 9, significantly improved the detection accuracy for multicolumn, nested, and irregular tables.

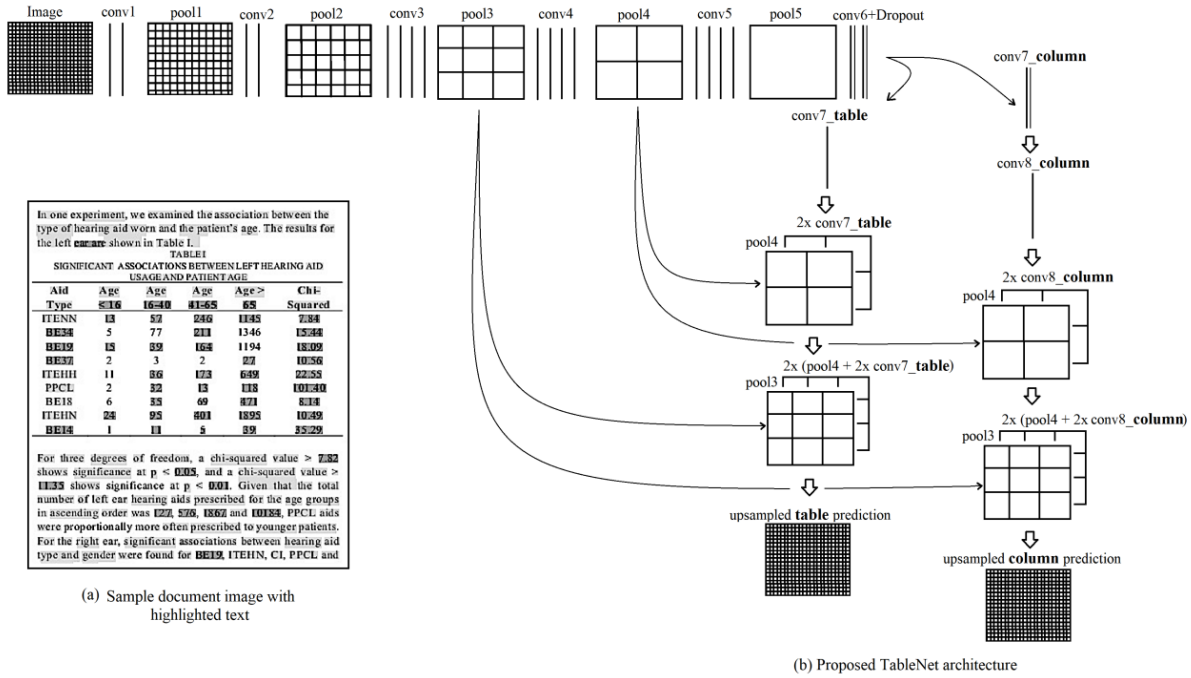


Figure 9: TableNet architecture proposed by Paliwal et al. [8].

Open-source tools such as Camelot and Tabula can also be used to detect and extract tables from text-based PDFs. However, due to their rule-based nature, these tools often fail to perform accurately on complex or densely populated tables, such as those in scientific datasheets.

In summary, while traditional rule-based methods paved the way for detecting tables present in documents, DL, and transformer-based architectures present a more robust solution to handle complex and irregular tables with varying styles.

2.2 Table Structure recognition

Table Structure Recognition (TSR), previously known as Table Understanding, is a key step in document processing that involves accurately identifying the structure of tables, including their rows, columns, headers, and merged cells. It is crucial in transforming unstructured tables into

structured data formats. TSR focuses on detecting and identifying internal table components and understanding how these components relate to each other.

Early TSR research relied on rule-based and heuristic approaches that inferred table structure from visual cues such as line segmentation, alignment, and whitespace. Itonori [9] proposed a heuristic-based approach that involved using the bounding boxes of textblocks to recognize the structure of partially ruled tables by expanding the textblock regions as seen in Figure 10. Kieninger [35] proposed a method based on geometric heuristics that expands a seed logical text block to all words that interleave with their vertical neighbors thereby, forming column blocks. The individual cells were then derived by conjoint decomposition of these column blocks.

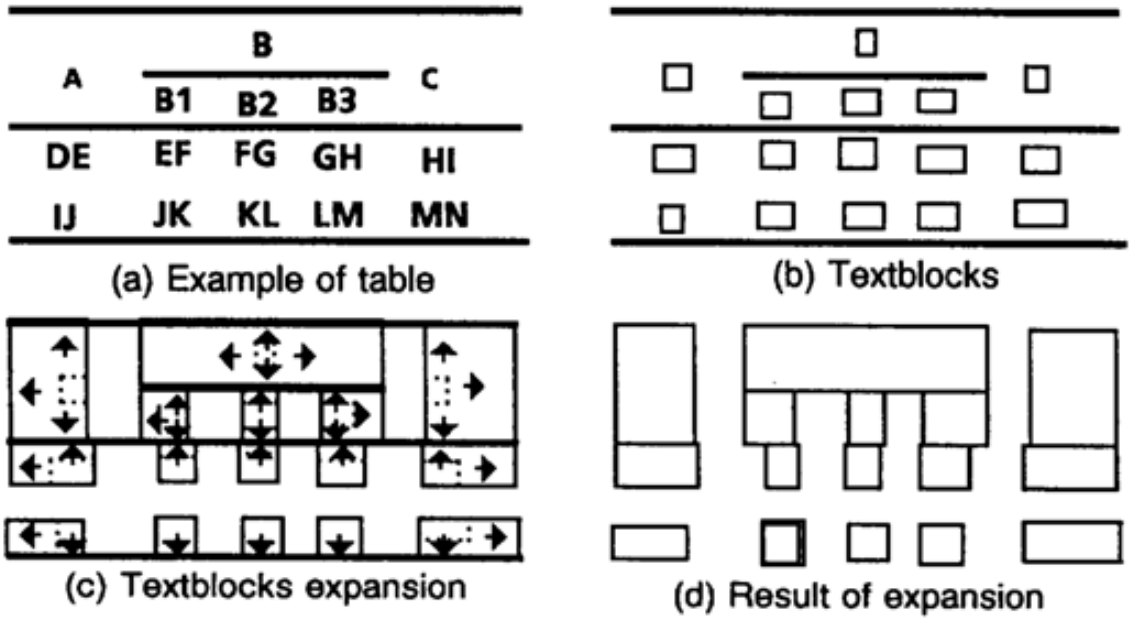


Figure 10: Approach of expanding textblocks to align with ruled line for TSR proposed by Itonori [9].

Rule-based methods suffered as document complexity increased, prompting researchers to explore Machine Learning and Deep Learning solutions that offered more flexibility and adaptability. Convolutional neural networks (CNNs) gained popularity in TSR tasks due to their ability to learn spatial hierarchies in images. Schreiber et al. [34] proposed an approach to classify table cells and identify their relationship using R-CNNs. This significantly improved the model's ability to identify complex structures and set a new baseline for DL-based Table Structure Recognition tasks.

Recently, transformer models like Table Structure Recognition with Transformers (TSRFormer) proposed by Li et al. [10] from Microsoft, have excelled in recognizing complex and diverse table structures by utilizing a two-stage Detection Transformer (DETR) approach with self-attention

as seen in Figure 11. It captures cell dependencies in large tables and excels in identifying hierarchical structures, borderless cells, empty or spanning cells, multi-layer headers, and tables with geometric distortions. Similarly, Raja et al. [11] developed a DL-based object detector that focuses on accurately identifying empty cells to improve the recognition of multi-row or multi-column cells. Figure 12 depicts the approach proposed by Raja et al. [11].

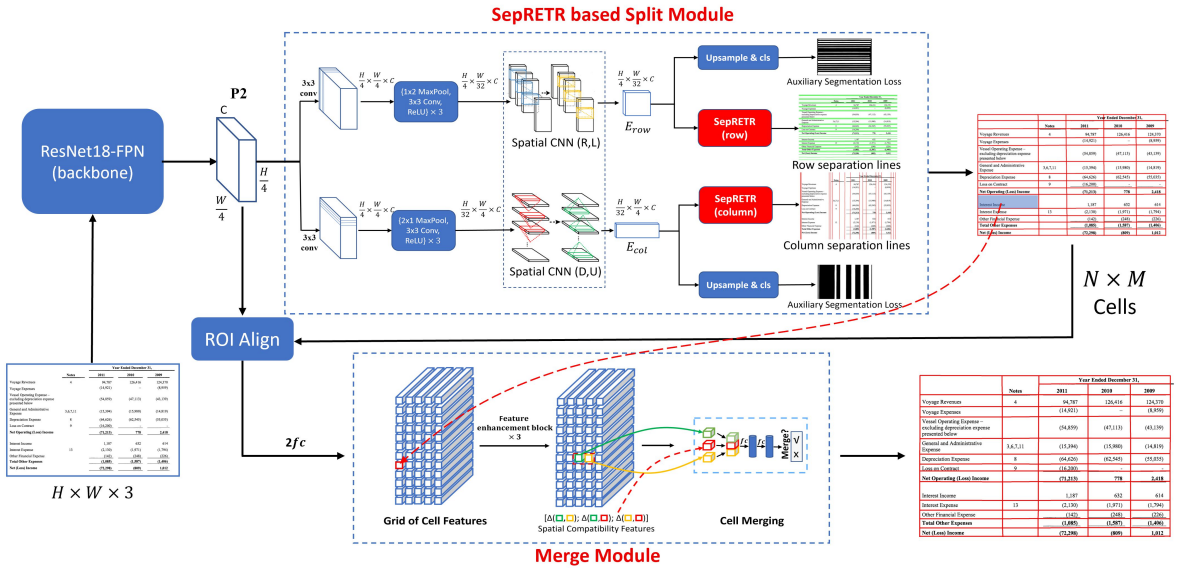


Figure 11: TSRFormer Architecture [10] - Transformer-based TSR model utilizing a split-merge approach for accurate table cell detection and segmentation.

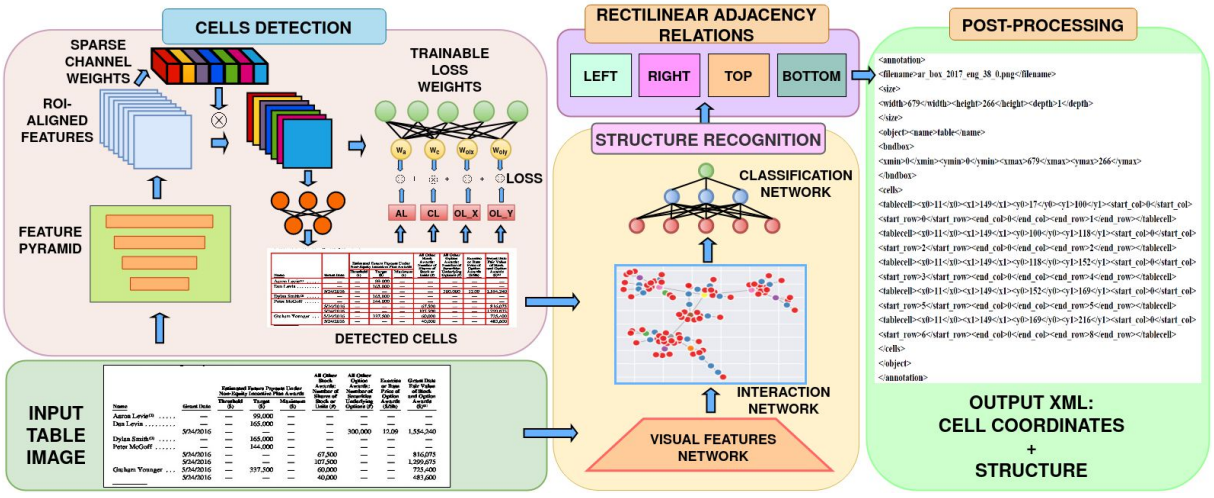


Figure 12: A multi-stage TSR approach proposed by Raja et al. [11] involving cell detection, adjacency-based structure recognition, and post-processing for extracting table structures.

Another breakthrough in Table Structure Recognition was achieved when Chi et al. [12] proposed GraphTSR, which represents table cells as graph nodes to capture spatial and structural

relationships among cells. By employing Graph Neural Networks (GNNs), GraphTSR learns the relationships between cells and rows, allowing it to perform well on tables with non-rectangular structures and merged cells. As seen in Figure 13, this method recognizes table structure through a four-step process. First, in the pre-processing stage (a), the cell contents are extracted along with their corresponding boundary boxes. Next, graph construction (b) is performed by building an undirected graph based on the detected cells. This is followed by a relation prediction step (c), where the proposed GraphTSR model predicts adjacent relationships between cells. Finally, the post-processing stage (d) reconstructs the table structure from the labeled graph. It also identifies intricate table relationships, making it valuable for complex document structures. Zhong et al. [36] proposed a similar approach that combined the benefits of CNN for visual feature extraction and graph networks for joint table detection and cell structure recognition tasks.

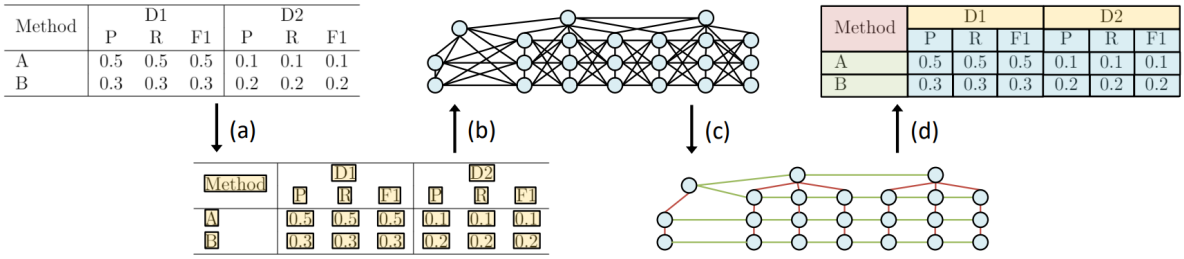


Figure 13: Graph-Based Table Structure Recognition proposed by Chi et al. [12].

Although DL-based methods perform well on unseen data, they still need a large amount of training data to achieve good performance, which is not readily available. Whereas transformer based models and GNNs are helpful where accurately extracting tabular data from complex documents is crucial due to their enhanced understanding of hierarchical and non-rectangular tables.

2.3 Existing Pipeline for PV data extraction using Lightning-Table

The Lightning-Table pipeline, introduced by Malik [13], uses a combination of DL and rule-based methods to extract data from Photovoltaic (PV) cell datasheets. This pipeline retrieves relevant tabular data from the datasheets and transforms them into structured data for further PV analysis. This pipeline also addressed the issue of information extraction from documents with diverse layouts.

The proposed pipeline was divided into three major stages, as seen in Figure 14. A DL-based object detection model was used to identify and localize tables within the datasheets. Then open-

source tools like Tabula [30] and Camelot [15] were used to recognize the structure and extract raw data from the detected tables alongside, a baseline method for performance comparison. However, these libraries could only extract textual content from text-based PDF files. Finally, a set of rule-based methods reformats the raw extracted data into a structured format, making them compatible with existing analytical tools for further analysis.

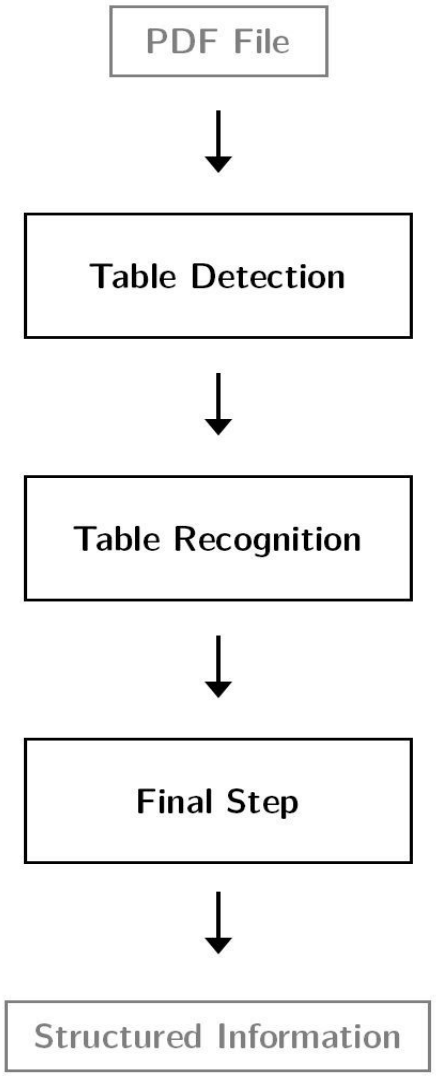


Figure 14: Workflow of the Table Extraction approach by Malik [13].

Figure 15 demonstrates an example of the Lightning-Table approach, where the tables are detected first, as indicated by the orange bounding boxes. Then, the raw values are extracted. Finally, the required values are structured and validated. The column *'EFF Code(%)'* is mapped to *'Efficiency'* and the column *'Pmpp(W)'* is mapped to *'Power @ Max Power Point'* using rule-based methods.

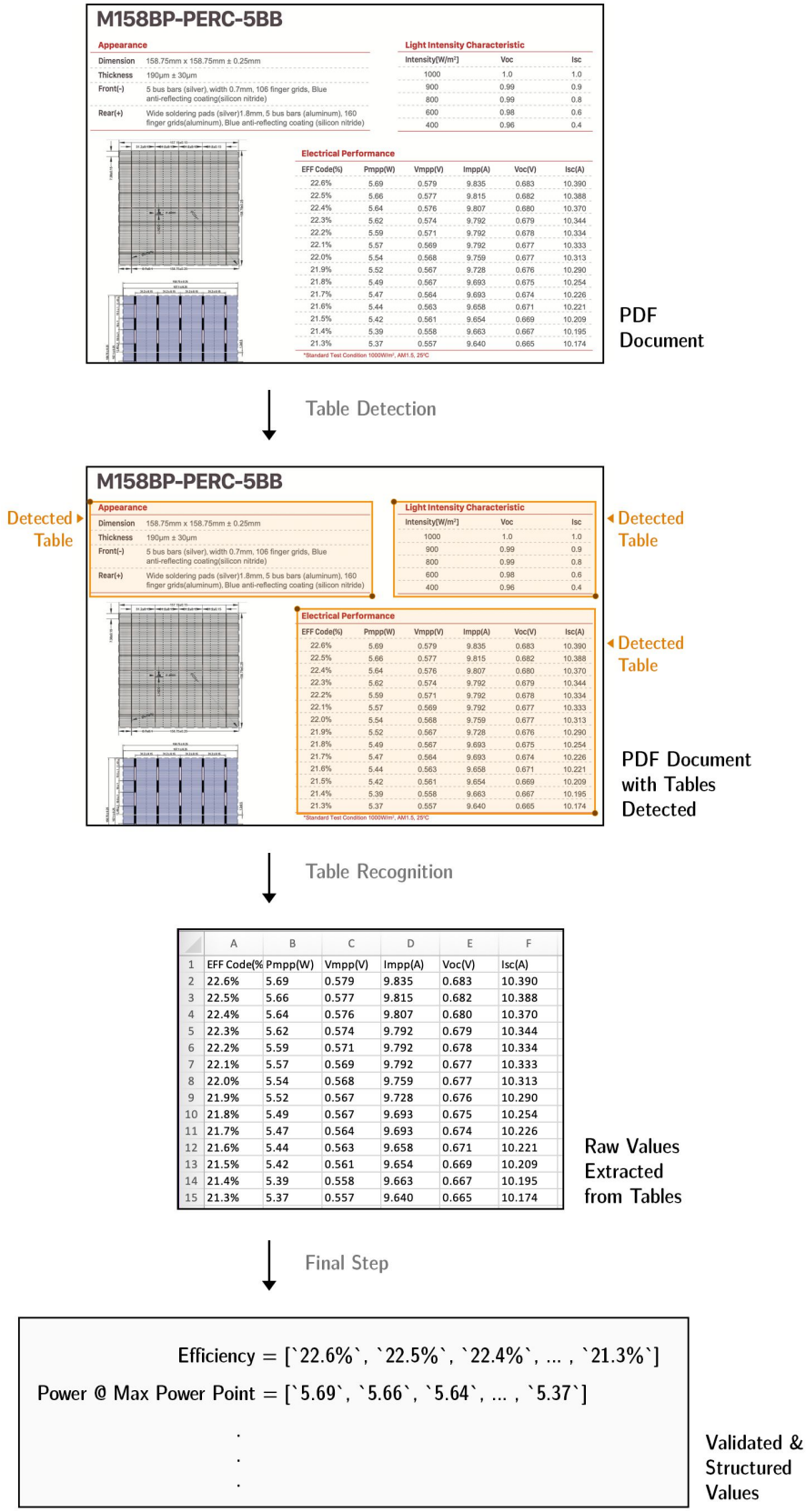


Figure 15: Table extraction process proposed by Malik [13] for converting unstructured tabular data from PDFs into structured information.

2.3.1 Drawbacks of Lightning-Table

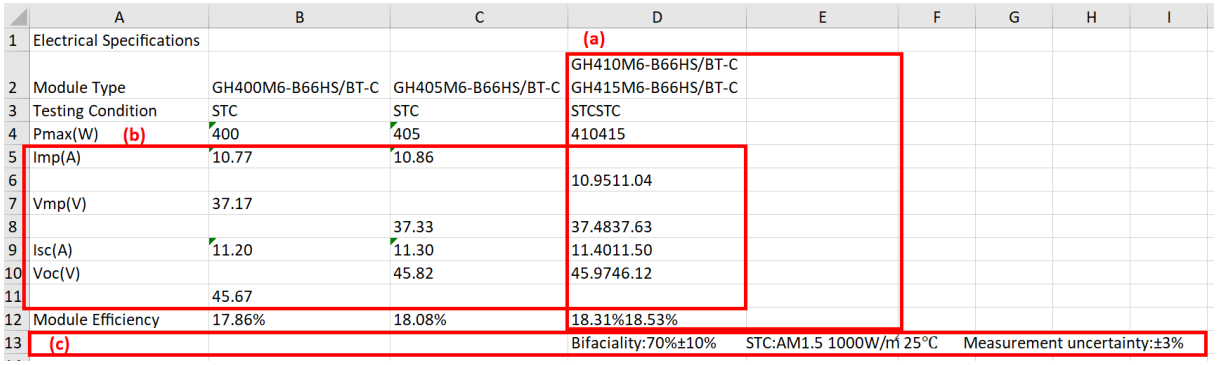
Although the Lightning-Table pipeline works well in extracting data from simple tables present in PV cell datasheets, it struggles to deal with complex table structures, resulting in significant drawbacks. For example, when tables contain merged cells, variable column spans, or complex table structures, the data extraction quality suffers, resulting in partially or incorrectly extracted data that hinders the reliability of analysis based on such outputs. Data extraction tools like Camelot [15] and Tabula [30] also consider each line separately, resulting in fragmented entries in the case of multi-line text cells. This reduces data continuity and can complicate data aggregation. Furthermore, they are primarily designed for text-based PDFs and cannot work on image-based or scanned PDF documents, thereby limiting the scope of the pipeline.

Inconsistencies in text alignment and line spacing in the documents can lead to incorrect data parsing and organization, as rule-based extraction cannot handle these variations. Also, the inconsistent naming conventions across manufacturers made identifying and categorizing relevant data difficult despite using regular expressions to handle this variability. Although ML and rule-based methods were applied for validation, the pipeline could not handle complex validation scenarios containing complex tables with multiclass classifications, nested tables, or implicit data, which would require a much deeper semantic understanding.

These drawbacks resulted in some deficiencies such as missing values in the extracted tables, incorrect merging of data, or improper table structure. Figure 16 depicts an example of a table from a PV cell datasheet and Figure 17 illustrates the data extracted from the table in Figure 16 using the Lightning-Table approach with Camelot [15] for data extraction. In the example in Figure 17 (a), the data from the last two columns are merged together. Missing and misaligned values can also be seen in Figure 17 (b) due to varying line spacing. Finally, the inability of Camelot [15] to represent merged cells can be observed in Figure 17 (c).

Electrical Specifications				
Module Type	GH400M6-B66HS/BT-C	GH405M6-B66HS/BT-C	GH410M6-B66HS/BT-C	GH415M6-B66HS/BT-C
Testing Condition	STC	STC	STC	STC
Pmax(W)	400	405	410	415
Imp(A)	10.77	10.86	10.95	11.04
Vmp(V)	37.17	37.33	37.48	37.63
Isc(A)	11.20	11.30	11.40	11.50
Voc(V)	45.67	45.82	45.97	46.12
Module Efficiency	17.86%	18.08%	18.31%	18.53%
		Bifaciality:70%±10%	STC:AM1.5 1000W/㎡ 25℃	Measurement uncertainty:±3%

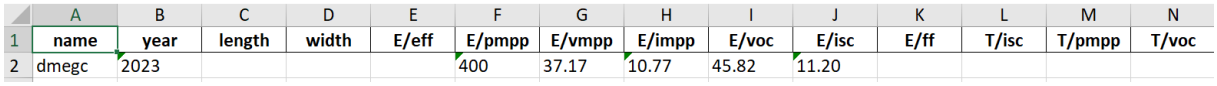
Figure 16: Simple table capturing the Electrical Specifications of a PV module [14].



	A	B	C	D	E	F	G	H	I
1	Electrical Specifications			(a)					
2	Module Type	GH400M6-B66HS/BT-C	GH405M6-B66HS/BT-C	GH410M6-B66HS/BT-C GH415M6-B66HS/BT-C					
3	Testing Condition	STC	STC	STCSTC					
4	Pmax(W) (b)	400	405	410415					
5	Imp(A)	10.77	10.86						
6				10.9511.04					
7	Vmp(V)	37.17							
8			37.33	37.4837.63					
9	Isc(A)	11.20	11.30	11.4011.50					
10	Voc(V)		45.82	45.9746.12					
11		45.67							
12	Module Efficiency	17.86%	18.08%	18.31%18.53%					
13	(c)			Bifaciality:70%±10%	STC:AM1.5 1000W/m 25°C		Measurement uncertainty:±3%		

Figure 17: Data extracted from Figure 16 using Lightning-Table with Camelot [15], highlighting extraction inconsistencies

Consequently, the rule-based data extraction and reformatting in the final step of this pipeline will fail as a result. Figure 18 depicts the output of the final data extraction and reformatting step. It can be observed that not all data was captured, and that there are many missing values. However, improving the tabular data extraction in the previous step should significantly improve these results.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**name**	**year**	**length**	**width**	**E/eff**	**E/pmpp**	**E/vmpp**	**E/impp**	**E/voc**	**E/isc**	**E/ff**	**T/isc**	**T/pmpp**	**T/voc**
2	dmegc	2023				400	37.17	10.77	45.82	11.20				

Figure 18: Final output of the Data Extraction process illustrating missing values.

3 Background

This chapter briefly introduces the fundamental concepts explored in the thesis, including the underlying technologies and models utilized.

3.1 Machine learning

Machine Learning (ML) algorithms predict the outcomes of unseen scenarios by learning patterns and making decisions based on historical data rather than explicit programming. Unlike rule-based systems, ML allows systems to adapt and generalize thereby, solving problems beyond the predefined scenarios. Machine Learning can be classified into two types: Supervised and Unsupervised Learning. Supervised Learning involves training a model on labeled data, where each input corresponds to a known output. The model then adjusts its weights and parameters iteratively during training to minimize the error between the model predictions and the predefined outputs to allow the model to improve its accuracy with more feedback and better generalize to unseen data. On the other hand, Unsupervised Learning identifies patterns and structures in unlabeled data without predefined outputs. This thesis uses Supervised Learning for the Table Classification task.

3.2 Deep Learning

Deep learning is a subset of Machine Learning with multiple hidden layers to learn from large datasets, and this section introduces the various architectures used in this pipeline and advanced methodologies like transfer learning and hyperparameter optimization, that were used to optimize the model performance.

3.2.1 Foundation of Deep Learning

Deep learning is based on Artificial Neural Networks (ANNs), which is inspired by the structure and operation of the human brain. These networks are made up of layers of interconnected nodes

called neurons. These nodes compute a weighted sum of their inputs and apply a non-linear activation function, as seen in Formula 1.

$$z = b + \sum_{i=1}^{n} w_i x_i, \quad y = \sigma(z) \tag{1}$$

Formula 1: Weight computation of neurons

Here, $\boldsymbol{w_i}$ are the *weights*, $\boldsymbol{x_i}$ are the *inputs*, $\boldsymbol{b}$ is the *bias*, and $\boldsymbol{\sigma(z)}$ is the *activation function*.

Figure 19 depicts a Feed-Forward Network (FNN) with input, hidden, and output layers, where each neuron is connected through weighted links. These hidden layers enable the network to learn increasingly abstract representations as the data propagates through the layers.

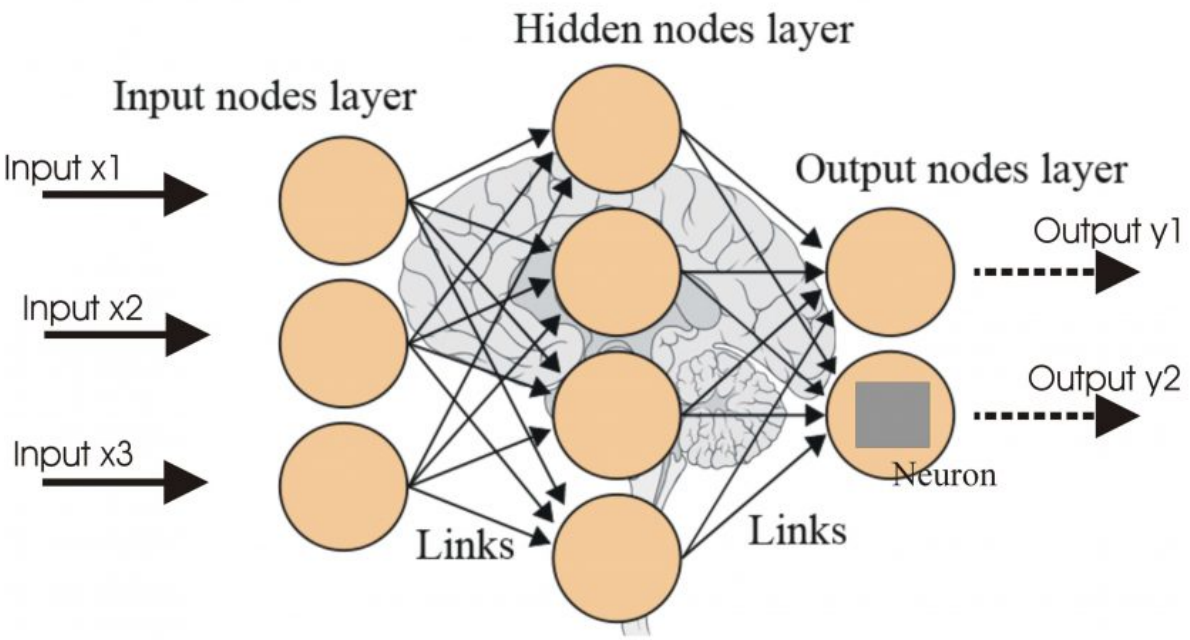


Figure 19: Fully-connected Feed-Forward Network [16].

3.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) were designed explicitly for processing grid-like data structures such as images. They use convolutional layers, which apply learnable filters over the input image to produce feature maps to extract spatial features hierarchically. This is followed by a pooling layer that down-samples the feature maps in spatial dimensions to reduce computational

complexity while preserving vital features, as seen in Figure 20. Finally, a set of fully-connected layers with a probabilistic distribution function recognizes and classifies objects. CNNs start by extracting simple features such as edges and textures in the early layers of the network and progress to more abstract shapes and patterns in deeper layers.

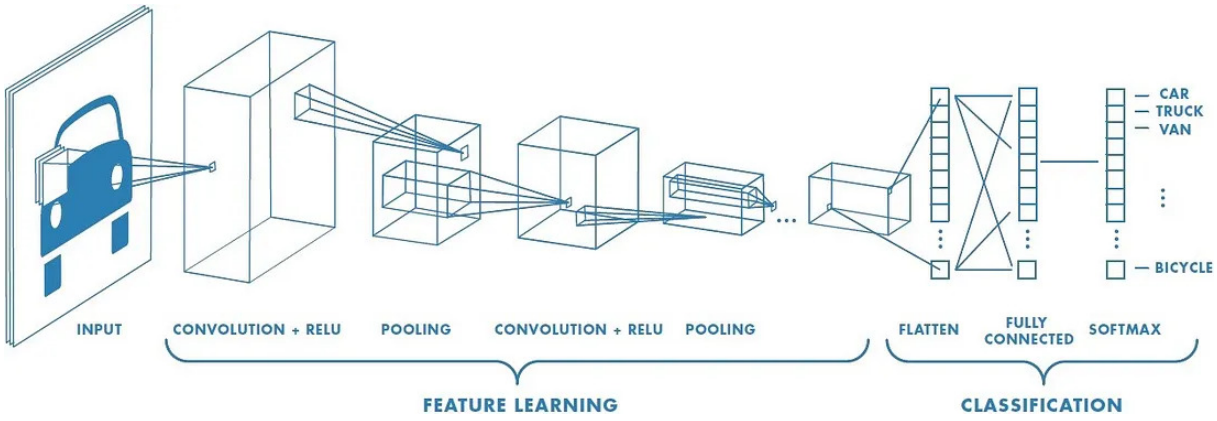


Figure 20: CNN architecture for object detection and classification [17].

CNNs ability to recognize spatial patterns can be leveraged to detect and recognize the structure of tables in PV module datasheets, where arrangements of rows and columns could vary significantly.

3.2.3 Residual Networks

Residual Networks (ResNets) are a type of Deep Neural Network (DNN), designed to address the problem of vanishing and exploding gradients in deep networks. As the depth of neural networks increases, the networks may fail to converge or degrade in performance due to vanishing or exploding gradients that arise from repeatedly multiplying gradient values across layers. Gradients below one vanish and stall learning, while those above one explode and destabilize training. This causes suboptimal performance or prevents network convergence.

To overcome this issue, Zhang et al. [18] introduced ResNets which can skip connections by bypassing one or more layers of the network. Figure 21 depicts the building block of Residual Networks. Each residual block consists of a few convolutional layers with batch normalization and ReLU activations, plus a skip connection. This design tackles vanishing gradients by letting the network learn residual functions instead of unreferenced mappings, making it easier to train and enabling deeper architectures that perform at least as well as their shallower counterparts.

Unlike typical CNNs which learn the direct mapping H(x), Residual Blocks introduce identity mappings through skip connections, which learn the residual function. From Formula 2, the

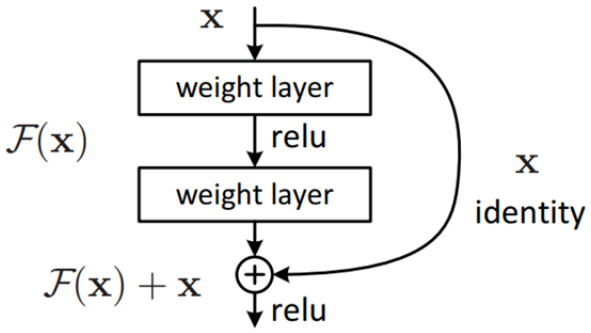


Figure 21: Residual block where the input is bypassed through an identity connection [18].

residual function is defined as the difference between the direct mapping and the input. The final output of a residual block enables identity mapping by adding the input to the residual function as observed in Formula 3.

$$F(x) = H(x) - x \tag{2}$$

Formula 2: Residual function computation

where, $\boldsymbol{x}$ is the *input to the block* and $\boldsymbol{F(x)}$ is the *residual function.*

Hence, the final output is computed as:

$$y = F(x) + x \tag{3}$$

Formula 3: Output of the residual block

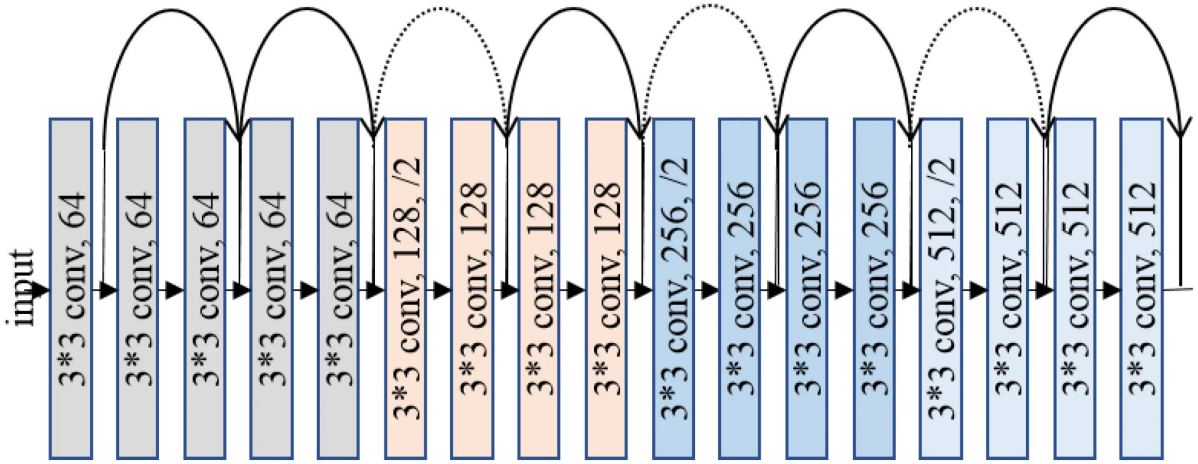


Figure 22: ResNet architecture [19].

Residual Networks have become the standard backbone for feature extraction, including those used for object detection and table recognition tasks. Figure 22 depicts a typical ResNet architecture.

Their ability to learn hierarchical feature representations efficiently is crucial for processing complex document layouts, such as those in PV module datasheets.

3.2.4 Transfer Learning and Fine-Tuning

Transfer learning allows the application of pre-trained models to new tasks with minimal training. Figure 23 demonstrates the difference between training a CNN from scratch and using a pre-trained model for transfer learning. Fine-tuning a Deep Learning model involves several crucial steps in adapting the pre-trained networks to perform a domain-specific task [20]. Firstly, layer freezing is employed, where the early layers of the network are frozen to retain the general features learned from pre-training on a larger dataset. This ensures that the foundational knowledge, such as basic edge or texture detection, remains intact. This is followed by selective updating, which fine-tunes the higher layers of the network to adapt to domain-specific patterns. This approach reduces computational costs and training time while enhancing model accuracy.

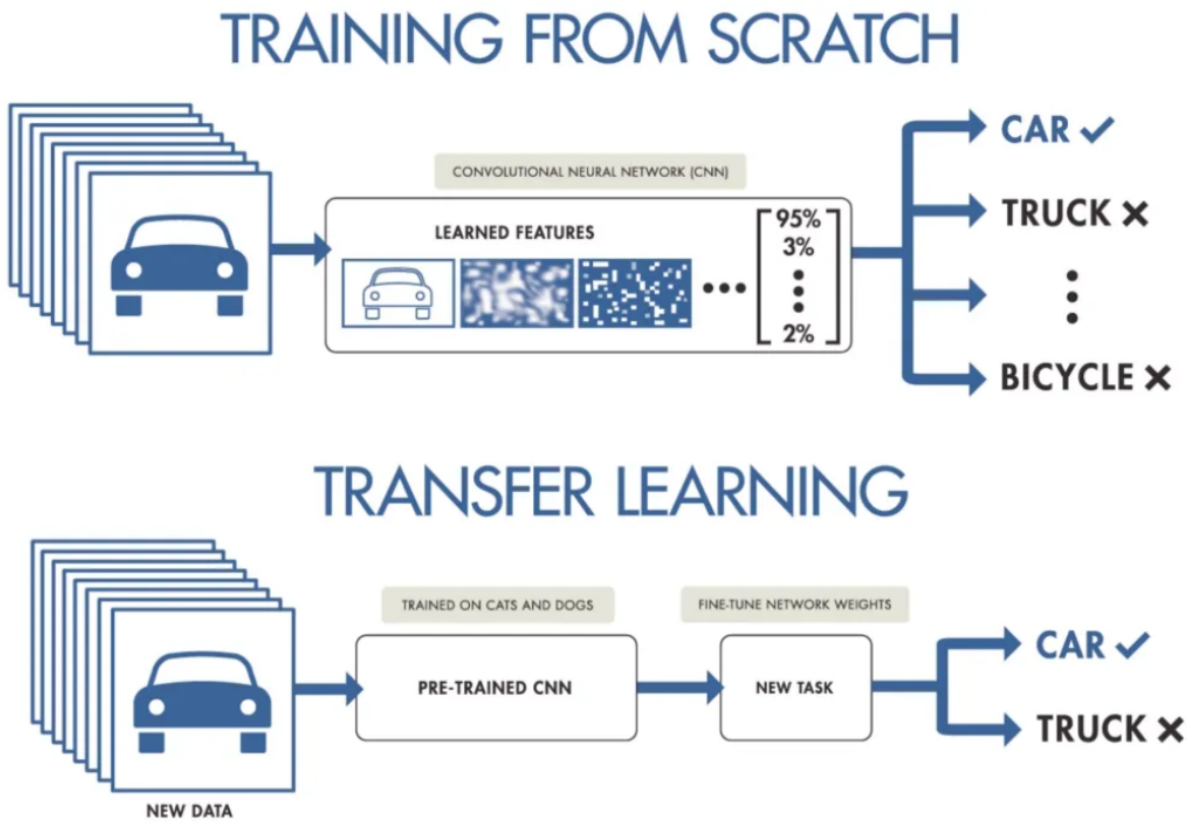


Figure 23: Illustration of Transfer Learning vs. Training from Scratch [20].

3.2.5 Hyperparameter Optimization

Deep learning models learn their internal parameters independently through the process of gradient descent, which adjusts the model weights to minimize error during training. However, some high-level parameters remain under the user's control and greatly impact the training dynamics and performance of the models. The following hyperparameters shape the learning process to improve the model's accuracy. They are as follows:

- **Learning Rate :** It determines the step size during weight updates. If the learning rate is too small, the network learns at a slow pace, prolonging the model convergence. On the other hand, if the learning rate is too high, it causes erratic updates and prevents the network from minimizing the loss.
- **Batch Size :** It regulates the amount of data that is processed at a time during training. This affects the memory usage and the accuracy of gradient estimation.
- **Network Depth :** As the depth of the network or the number of neurons per layer increases, the network becomes complex, enabling the model to capture more intricate patterns and reduce underfitting (which occurs when the model is too simple to capture the underlying patterns in the data). Conversely, reducing layers can simplify the network and mitigate overfitting (which occurs when the model learns the training data too well, including noise and random fluctuations. The model then performs exceptionally well on the training data but poorly on unseen or new data).
- **Regularization :** Techniques, such as dropout, prevent model overfitting by deactivating a subset of neurons during training. This helps the model generalize on unseen data.
- **Weight Decay :** It regularizes model parameters by penalizing large weights, preventing model overfitting, and improving generalization.
- **Learning Rate Scheduler :** It dynamically adjusts the learning rate during training, promoting faster convergence and avoiding local minima.
- **Number of Queries :** Sets a limit on the number of function evaluations or training runs to balance exploration of hyperparameter space and computational cost.
- **Early Stopping :** This stops the model training when performance stops improving, preventing overfitting, ensuring better generalization and saving computational resources. Figure 24 shows the criteria for early stopping.

By optimizing these hyperparameters, the DL model learns and adapts to specific tasks.

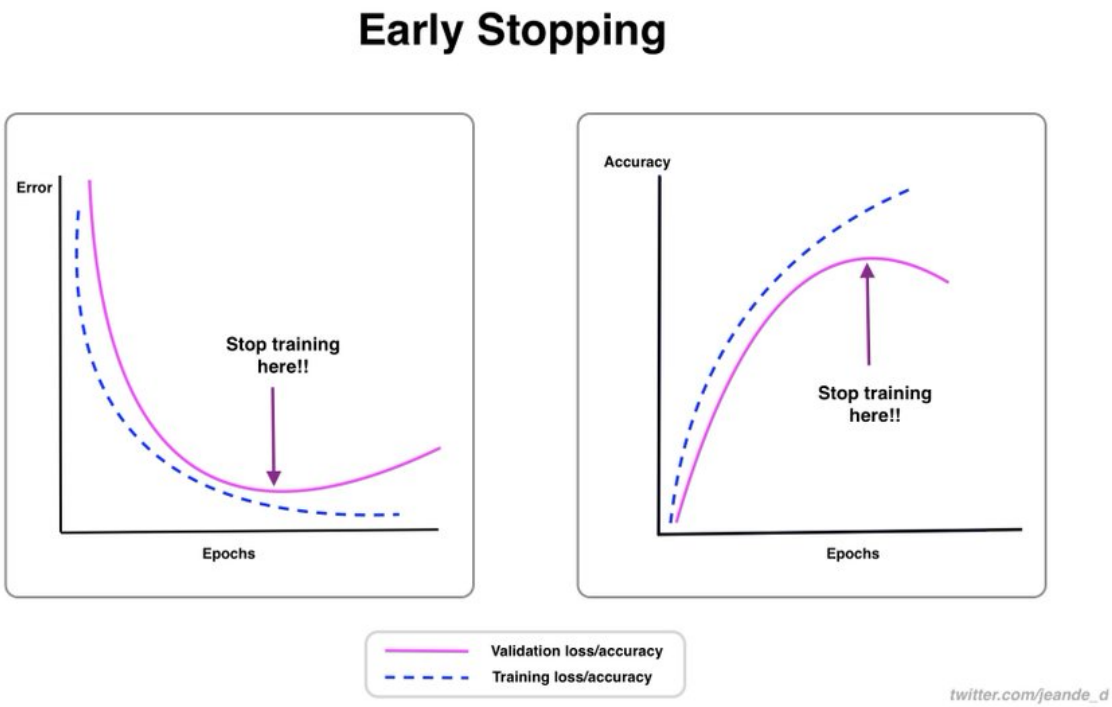


Figure 24: Early stopping criteria to halt training when validation loss starts increasing [21].

3.3 Object Detection

Object detection is a field of computer vision that deals with identifying and localizing objects in an image. Object detection techniques were applied on the PV module datasheets to identify tables and their internal components.

3.3.1 Detection Transformers

Detection Transformer (DETR) proposed by Carion et al. [22] from Facebook is a transformer based architecture developed for object detection tasks. It employs self-attention mechanisms to model long-range spatial dependencies in images more effectively than CNNs. This is useful in processing document images such as tables, charts, and text blocks with complicated spatial relationships.

The DETR model views object detection as a set prediction problem, eliminating the need for region proposal networks and anchor boxes [22]. Its architecture is based on a CNN backbone, which extracts spatial features from the input image to form the base representation, as observed in Figure 25. These features are then passed to a transformer encoder, which embeds them with positional encodings that capture spatial relationships. Then, a decoder is equipped with learnable

object queries to predict both the bounding box and the class label. The transformers multi-head self-attention mechanism models interactions between image regions and queries, enabling precise localization and classification. DETR uses bipartite matching (one-to-one correspondence) at training to uniquely associate predicted bounding boxes and ground-truth objects, thus providing a precise and unambiguous prediction. This transformer-based model directly predicts the locations and classes of objects, enabling it to be more efficient in dense layouts with overlapping objects.

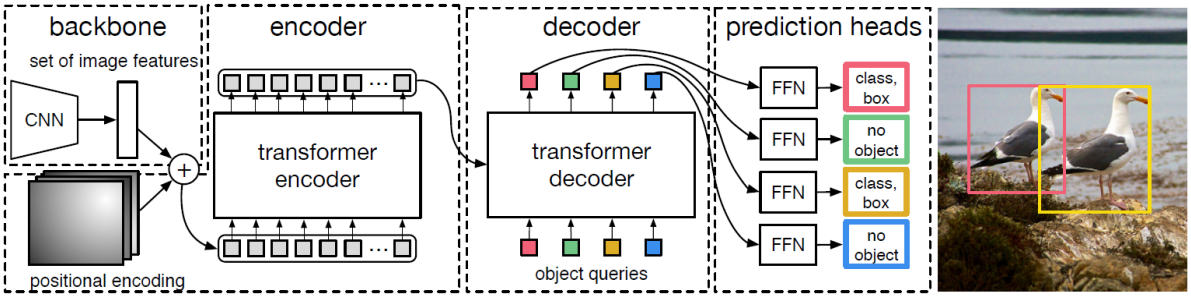


Figure 25: DETR architecture featuring a backbone, a transformer encoder, a transformer decode, and its prediction heads [22]

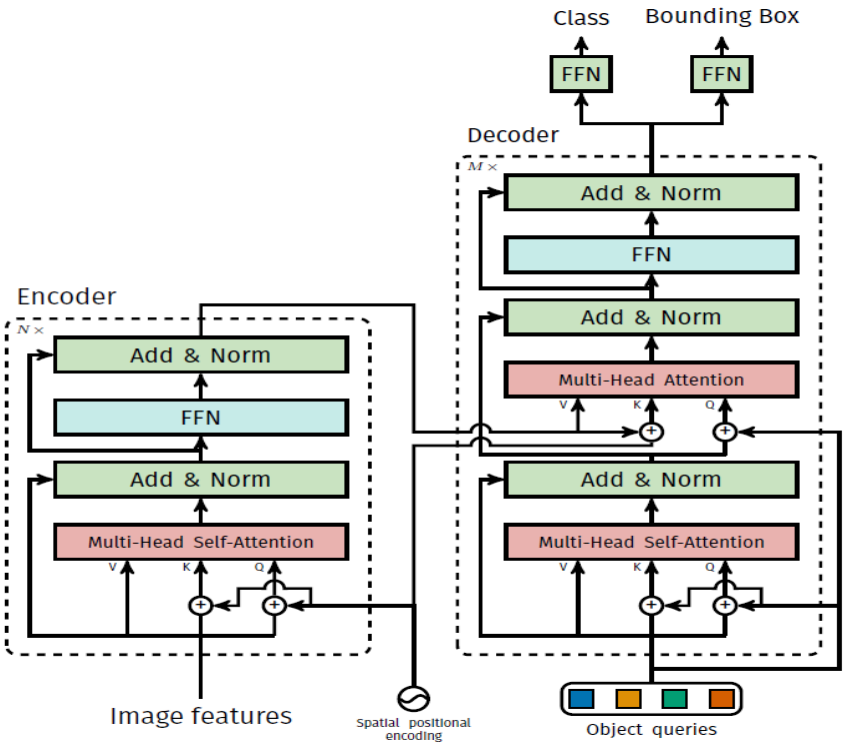


Figure 26: Encode-Decoder Architecture of DETR [22]

Figure 26 depicts how encoder inputs, which consist of image features combined with spatial positional encoding, are added into the queries and keys at each multi-head self-attention layer.

Then, the decoder receives queries (initially set to zero), output positional encoding (object queries), and encoder memory to generate the final set of predicted class labels and bounding boxes. This process occurs in parallel through multiple multi-head self-attention and decoder-encoder attention mechanisms. However, this requires high computational resources (especially for high-resolution images) and relies on a fixed number of object queries, which can lead to inefficiencies if the actual number of objects does not match the preset query count.

3.4 Text Classification

Text classification is a crucial post-processing step in document analysis which organizes the text content into predefined categories and allows for targeted processing of only the essential sections within each document.

3.4.1 Word Vectorization

Unlike humans, computers cannot extract features from text-based documents using words themselves as features and require additional quantification of words to understand the data. Word vectorization is a technique that transforms textual data into numerical vectors that denote semantic meaning, making it easier for Machine Learning models to interpret language-based information. It not only measures word usage, but also creates a comprehensive vocabulary from all documents, giving each word its proper context. The word vectors lay the groundwork for further categorization, allowing integration with classification models and boosting the precision of data retrieval from complex documents.

Term Frequency-Inverse Document Frequency (TF-IDF)
TF-IDF quantifies the importance of a word in a document for any textual document collection. The term-frequency is obtained by dividing the frequency of appearance of that term in the document by the frequency of all terms in that document [37]. The basic form of TF-IDF is described below mathematically [37]:

$$tf(t,d) = \frac{f_{t,d}}{-\sum_{t \in d} f_{t,d}} \tag{4}$$

Formula 4: Formula to compute Term-Frequency

Where, $\boldsymbol{t}$ is a *term*, $\boldsymbol{D}$ is a *set of documents*, $\boldsymbol{d}$ is a *document in the set of documents (D)*, $\boldsymbol{f}$ is the *raw count for the term*, and $\boldsymbol{N}$ is the *total number of documents.*

$$idf(t, D) = \frac{N}{|d \in D : t \in d|} \tag{5}$$

Formula 5: Formula to compute Inverse Document-Frequency

Finally, the TF-IDF is computed by multiplying the two terms.

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{6}$$

Formula 6: Formula to compute Term Frequency-Inverse Document Frequency

As the frequency of occurrence of a word in a document increases, the Term Frequency (TF) value increases, whereas, the Inverse Document Frequency (IDF) keeps track of the occurrence of the word in the document collection. If a word appears frequently across the document collection then, its significance in classifying the given text decreases. This addresses the fact that some words may appear often in a given set of documents.

3.4.2 Naive Bayes Classifier

Naive Bayes is a probabilistic Supervised ML model based on the Bayes Theorem which assumes that the features are independent of each other. For the task of Table Classification in this thesis, a Multinomial Naive Bayes (MNB) classifier was used to deal with multiple output classes. It is described mathematically as follows [38].

The probability of the document belonging to that class is given as:

$$P(c \mid d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k \mid c) \tag{7}$$

Formula 7: Formula for Multinomial Naive Bayes classification

Where, $\boldsymbol{c}$ is a *class*, $\boldsymbol{d}$ is a *document*, $\boldsymbol{P(t_k \mid c)}$ is the *probabilistic evidence that the term* t_k *belongs to class c*, and $\boldsymbol{P(c)}$ is the *background frequency of the class c.*

P(c) is interpreted as the probability of the class occurring relative to other classes. If the terms $\boldsymbol{t_k}$ 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 P(c) and the probability for the terms are learned from the training dataset, as defined by H. Schütze [38].

The background frequency P(c) is given by:

$$P(c) = \frac{N_c}{N} \tag{8}$$

Formula 8: Formula for computing the background frequency

Where, $\boldsymbol{N_c}$ is the *total number of documents in class c*, and $\boldsymbol{N}$ is the *total number of documents.*

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:

$$P(t \mid c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}} \tag{9}$$

Formula 9: Formula for computing the probabilistic evidence

3.5 Table Transformer

Table Transformer (TATR), introduced by Smock et al. [23], is a state-of-the-art solution to extract structured tabular data from document images. This pipeline performs three major tasks, as seen in Figure 27:

- **Table Detection**, where the tables within the document are identified and localized.
- **Table Structure Recognition**, which identifies the internal structure of the tables such as rows, columns, and cells.
- **Table Functional Analysis**, which recognizes the table's keys and values.

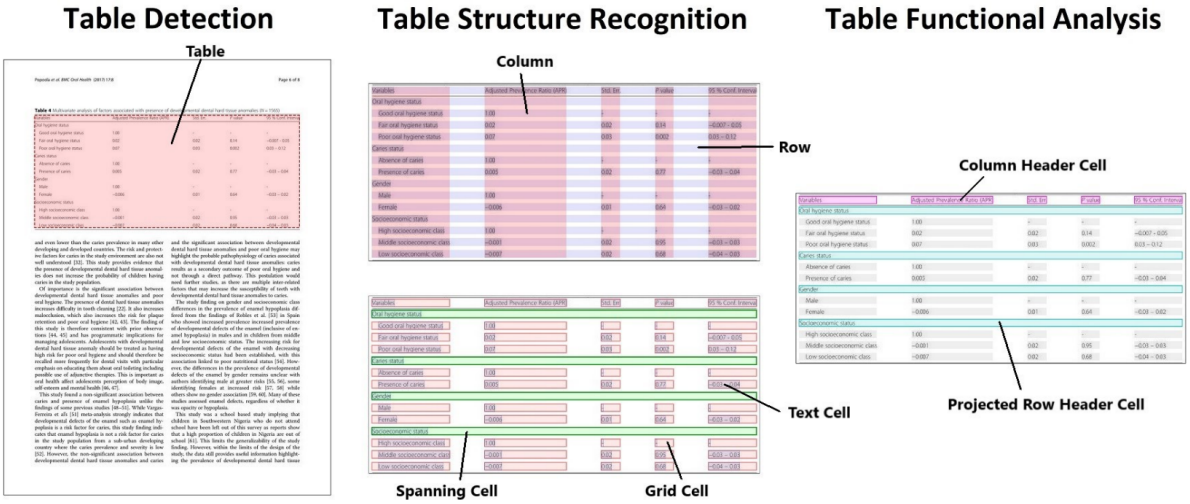


Figure 27: Subtasks addressed by Table Transformer [23].

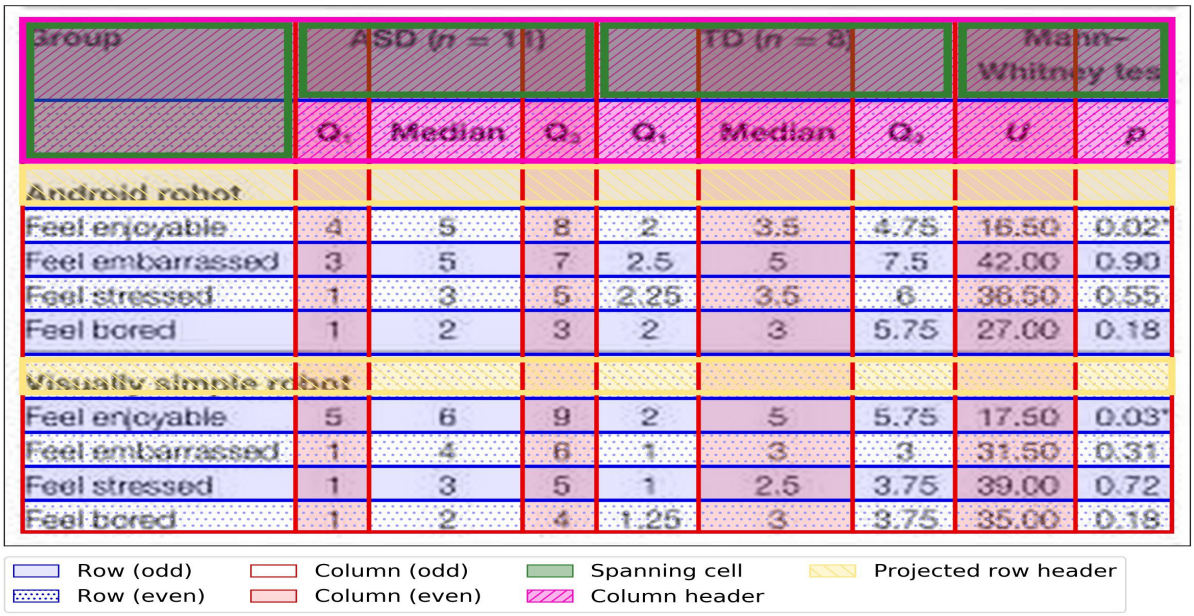


Figure 28: Bounding box annotations for various object classes in a table [23].

TATR uses two DETR models along with OCR-derived textual data to achieve high data extraction accuracy and adaptability across diverse document layouts. The detection model was trained to detect tables present in the document images. It was also trained to detect the rotated tables, that is, tables not aligned with the image axis, identify their degree of rotation, process them, and then return them to the normal image orientation. For training the structure recognition model, the table's hierarchical structure is modeled using six object classes: 1) *table*, 2) *table column*, 3) *table row*, 4) *table column header*, 5) *table projected row header*, and 6) *table spanning cell*. The representation of these object classes is illustrated in Figure 28. Additionally, the intersection of a pair of table column and table row objects forms an additional class called table grid cell.

3.5.1 Model architecture

The Table Transformer architecture (derived from the DETR architecture) is designed to detect and parse table structures in an end-to-end fashion by first extracting low-level visual features from the input document image using a ResNet backbone. These features are flattened, enriched with positional embeddings, and passed into a multi-head self-attention encoder to capture global relationships. The decoder then uses cross-attention to match queries (each representing a potential table or table component) with the encoded features. The model then applies a set-based prediction approach, leveraging a bipartite matching loss to enforce a one-to-one mapping between predicted and ground-truth structures, ensuring clean and non-overlapping table boundaries. Even without separate region proposals or extensive post-processing, this pipeline can handle complex tables, diverse layouts, and noisy inputs effectively through global attention and set-based matching.

3.5.2 Canonicalization Algorithm

Canonicalization is a crucial step in Table Structure Recognition (TSR) that solves the over-segmentation problem within table structure annotations. This is achieved by merging adjacent cells under specific conditions to ensure that the rows and columns align with the logical structure of the table while maintaining visual consistency. The recognized structures define how these elements are processed in later stages. In addition, multiple quality control steps were also implemented here.

Figure 29 depicts a table with an over-segmented structure annotation containing extra blank cells in the row and column headers. Whereas, the canonical structure annotation for the same table depicted in Figure 30 merges these cells and captures the table's true logical structure. The blank cells in the top-left corner are not part of the table and can be structured using any consistent scheme.

The canonicalization algorithm works on the assumption that all tables follow a consistent structure such as the Wang model [39], where the headers of the tables have a hierarchical structure that corresponds logically to a tree. Each cell in the structure corresponds to a node in the header tree, and data values are indexed clearly and uniquely. It ensures that the columns line up with the column header tree's leaf nodes, while the rows line up with those in the row header tree. This alignment allows for the exact modeling of a table's logical structure. The logic for each table structure is as follows:

		ΔSDM			
		better	**equal**	**Worse**	**Sum**
ΔSCA	**better**	19457 (28.9)	12 (0.02)	14654 (21.8)	**34,123 (50.8)**
	equal	1158 (1.7)	21989 (32.7)	1024 (1.5)	**24,171 (36.0)**
	worse	3755 (5.6)	2 (0.003)	5183 (7.7)	**8,940 (13.2)**
	Sum	**24370 (36.2)**	**22003 (32.7)**	**20861 (31.0)**	**67,234 (100.0)**

Figure 29: Over-segmented structure annotation example [23].

		ΔSDM			
		better	**equal**	**Worse**	**Sum**
ΔSCA	**better**	19457 (28.9)	12 (0.02)	14654 (21.8)	**34,123 (50.8)**
	equal	1158 (1.7)	21989 (32.7)	1024 (1.5)	**24,171 (36.0)**
	worse	3755 (5.6)	2 (0.003)	5183 (7.7)	**8,940 (13.2)**
	Sum	**24370 (36.2)**	**22003 (32.7)**	**20861 (31.0)**	**67,234 (100.0)**

Figure 30: Canonical structure annotation example[23].

- **Header Nodes:** The algorithm ensures that each node in the header tree has at least two children to avoid ambiguity. Every node with less than two children is adjusted according to the expected hierarchical structure.
- **Projected Row Headers (PRHs):** It extends the PRHs to other neighboring columns or rows until the tree's logical structure is consistent with the table structure as defined by Wang [39].
- **Spanning Cells:** Each spanning cell, including cells involving multi-row or multi-column headers, is refined to keep uniformity with Wang's model [39]. Special cases, like the rows of blank cells between parent and child nodes, are combined appropriately to preserve logical relationships.

Based on Wang's Model, Smock et al. [23] implemented the canonicalization algorithm depicted in Figure 31. The main steps involved in the canonicalization algorithm are as follows:

Addition of Cells to Headers:

- Blank spanning cells in the header are split into individual grid cells.

- Rows starting with blank cells are added to column headers.
- Rows labeled as part of the column header are expanded until all columns have at least one complete cell.

Merging of Cells:

- Header cells are recursively merged with adjacent cells in the column or row headers if they span identical ranges.
- Blank cells adjacent to header cells are merged together to maintain continuity, provided all adjacent cells are also blank.
- For projected row headers, all cells within the row are merged into a single entity.

Algorithm 1 PubTables-1M Canonicalization

ADD CELLS TO THE COLUMN AND ROW HEADERS
  Split every blank spanning cell into blank grid cells
  if the first row starts with a blank cell **then** add the first row to the column header
  if there is at least one row labeled as part of the column header **then**
    while every column in the column header does not have at least one complete cell that only spans that column **do:** add the next row to the column header
  end if
  for each row **do: if** the row is not in the column header and has exactly one non-blank cell that occupies the first column **then** label it a projected row header
  if any cell in the first column below the column header is a spanning cell or blank **then** add the column (below the column header) to the row header
MERGE CELLS
  for each cell in the column header **do** recursively merge the cell with any adjacent cells above and below in the column header that span the exact same columns
  for each cell in the column header **do** recursively merge the cell with any adjacent blank cells below it if every adjacent cell below it is blank and in the column header
  for each cell in the column header **do** recursively merge the cell with any adjacent blank cells above it if every adjacent cell above it is blank
  for each projected row header **do** merge all of the cells in the row into a single cell
  for each cell in the row header **do** recursively merge the cell with any adjacent blank cells below it

Figure 31: Canonicalization algorithm implemented by Smock et al. [23].

The recognized table structures were then subjected to a validation step that involved checking for consistency in the length of rows and columns to check if it forms a valid matrix. Irregularities, such as variable lengths, are discarded to avoid further processing errors and to ensure accurate table extraction.

4 Approach

This chapter outlines the end-to-end pipeline implemented for tabular data extraction from PV module datasheets. It expands on the techniques used to gather and prepare the data in Section 4.1 and the pre-processing of raw PDF documents in Section 4.2. The detailed methodology of locating tables and recognizing its internal table structures is presented in Section 4.3 and Section 4.4 respectively. Section 4.6 describes the tabular data extraction process and finally the post-processing of extracted information to gain useful insights is described in Section 4.7. Figure 32 depicts the intermediate stages of the proposed data extraction pipeline.

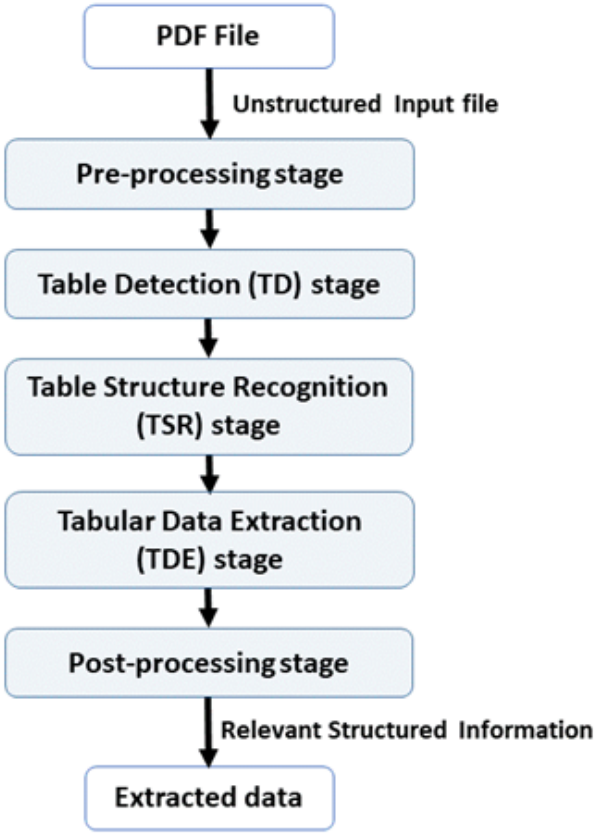


Figure 32: Overview of the proposed data extraction pipeline.

4.1 Data Collection and Data Preparation

The quality and quantity of the training data plays an important role in determining the performance and robustness of DL models. Given the diversity and complexity of the PV module

datasheets, this pipeline required advanced data preparation and augmentation techniques to maintain uniformity, handle variability, and enhance the model's generalization capability. This section describes the various processes involved in the collection, annotation, and augmentation of data for the Table Detection (TD) and Table Structure Recognition (TSR) tasks.

4.1.1 Solar Module Datasheets

Solar module datasheets provided by PV manufacturers provide technical specifications and performance metrics of PV modules. They include several tables in which the module's electrical, mechanical, and thermal properties are presented. However, these tables differ significantly in design, structure, and layout based on the manufacturer's branding scheme, product type, and document format. Hence the Data Extraction pipeline proposed for this thesis has to be robust and flexible to generalize over various table designs. Therefore, data collection, data annotation, and data pre-processing become crucial tasks.

4.1.2 Data Gathering and Data Annotation

There were no labeled datasets readily available for the Table Detection (TD) and Table Structure Recognition (TSR) tasks on PV module datasheets. Hence, two custom PV module datasets were created from publicly available datasheets for training and evaluating the Table Detection and Table Structure Recognition models respectively. The datasheets were directly downloaded from the official websites of PV manufacturers. The datasheets were sampled to cover a variety of manufacturers, table designs, complex structures, and presence of various terminologies. These datasheets, typically in PDF format, were then converted into images using the open-source software PyMuPDF, which served as the raw input for the subsequent data preparation stages [40]. The PDF images themselves served as inputs for training the Table Detection models, whereas the manually cropped tables from these images served as inputs for training the Table Structure Recognition model.

These images were then annotated using an open-source software called LabelImg, where bounding boxes were drawn manually around the regions of interest and labeled appropriately [41]. This manual approach introduced slight variations and alignment mismatch while annotating table elements which was then handled during post-processing. The annotation process slightly differed depending on the task at hand:

- **Table Detection (TD) :** Tables within the PDF images extracted from PyMuPDF [40] were annotated to train the detection model. To streamline the Table Classification task in the post-processing stage, the table header was also included within the table's boundary during annotation.
- **Table Structure Recognition (TSR) :** Table regions cropped from the PDF images were annotated by manually identifying and labeling their structural components, such as rows, columns, spanning cells, row headers, column headers, and table names. A cell could have multiple labels, which were resolved in the post-processing stage by prioritizing the most relevant ones.

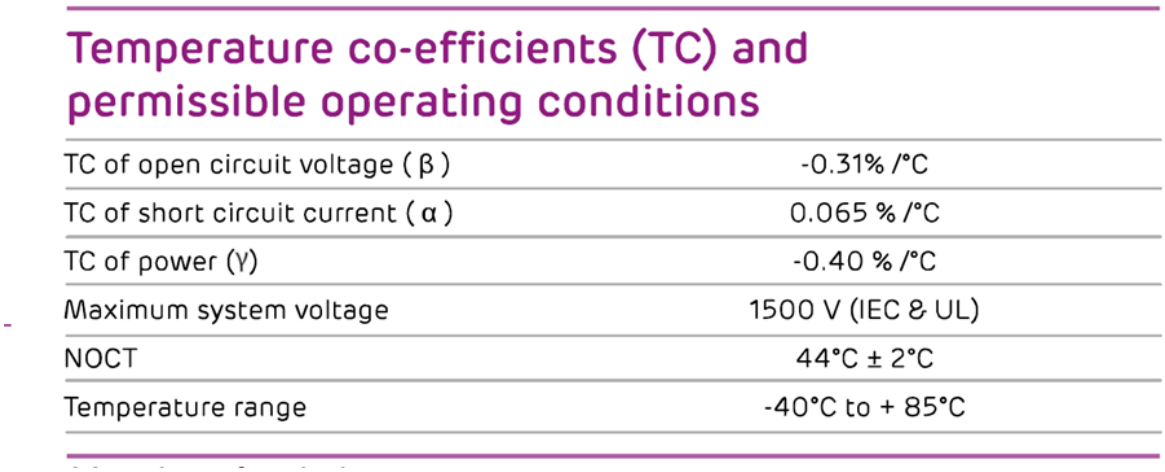
These annotations were then saved as JavaScript Object Notation (JSON) files containing each element's bounding box coordinates and associated class labels. These two custom PV datasets comprise diverse table layouts, ranging from simple single-row tables to complex multi-axis structures to capture real-world data complexity, such as inconsistent formatting, incomplete tables, and overlapping graphical elements.

4.1.3 Data Augmentation

Due to the gathered dataset's limited volume, data augmentation was implemented to expand the available data. The augmentation strategies proposed here included a variety of geometric and contextual transformations that aim to increase the representational power of the dataset. They include:

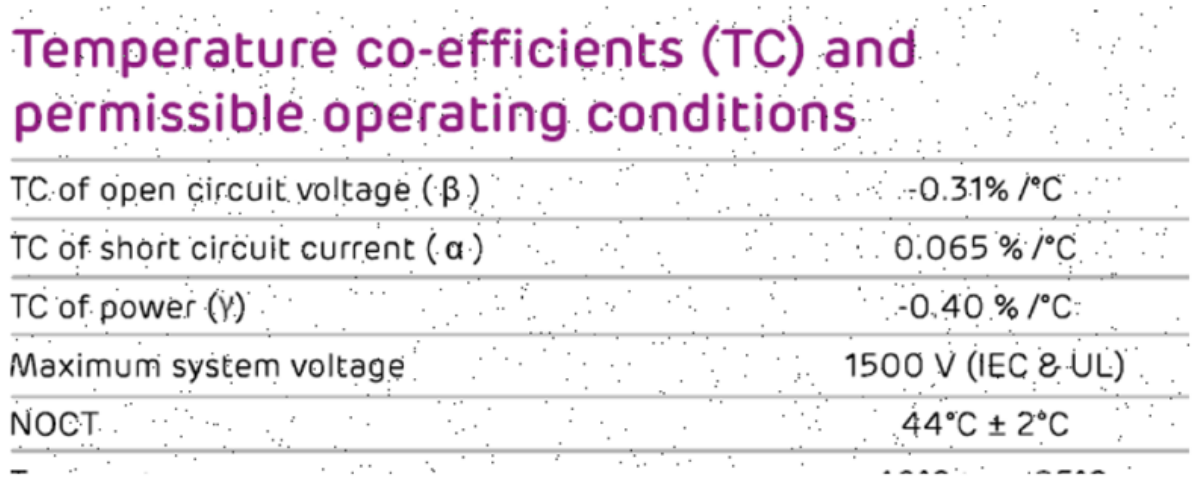
- **Geometric Transformations :** Rotations, scaling, and cropping were applied to mimic different orientations and perspectives of tables. This ensures that the model encounters various spatial arrangements.
- **Noise Injection :** Synthetic noise was added artificially to mimic the real-world artifacts found in low-quality datasheets. This helped the model become more resilient to imperfections in the document, like smudges, compression artifacts, watermarks, and faint text. Gaussian noise was added to simulate low-quality images whereas scanning imperfections were mimicked by randomly inverting or hiding pixels.
- **Color Adjustments :** Variations in brightness, contrast, and saturation helped simulate manufacturers' varying color schemes and scanning conditions.
- **Random Partial Masking :** Parts of the images were masked randomly with varying opacities to simulate occlusions or missing data, challenging the model to make inferences

based on incomplete visual information. This also helped in detecting tables with complex background colors or graphical elements.



Temperature co-efficients (TC) and permissible operating conditions	
TC of open circuit voltage (β)	-0.31% /°C
TC of short circuit current (α)	0.065 % /°C
TC of power (γ)	-0.40 % /°C
Maximum system voltage	1500 V (IEC & UL)
NOCT	44°C ± 2°C
Temperature range	-40°C to + 85°C

Figure 33: Example of a gathered input table image for training the Table Structure Recognition (TSR) model.



Temperature co-efficients (TC) and permissible operating conditions	
TC of open circuit voltage (β)	-0.31% /°C
TC of short circuit current (α)	0.065 % /°C
TC of power (γ)	-0.40 % /°C
Maximum system voltage	1500 V (IEC & UL)
NOCT	44°C ± 2°C

Figure 34: Augmentation of Figure 33 with a scaling factor of 0.63, crop ratio of 0.8, and Gaussian noise with mean=0 and variance=20.

Figure 34 was obtained by applying augmentation techniques such as scaling, cropping, and Gaussian noise injection on Figure 33. A random scaling factor of 0.63 was applied to the input image which reduced the image dimensions to 63% of its original size. Following this, the image was cropped with a random crop ratio of 0.80 which retained only 80% of the image area. This technique forces the model to detect structures within partial or incomplete views and helps simulate scenarios where parts of tables are missing due to poor Table Detection performance. Then, a Gaussian noise was injected with a mean of 0, which preserved the image's brightness and variance of 20 which controls the spread or intensity of the noise. This technique helps simulate

imperfections such as smudges or grainy textures in low-quality documents. Finally scanning artifacts were introduced by randomly inverting or hiding pixels to train models that are robust against real-world document degradation.

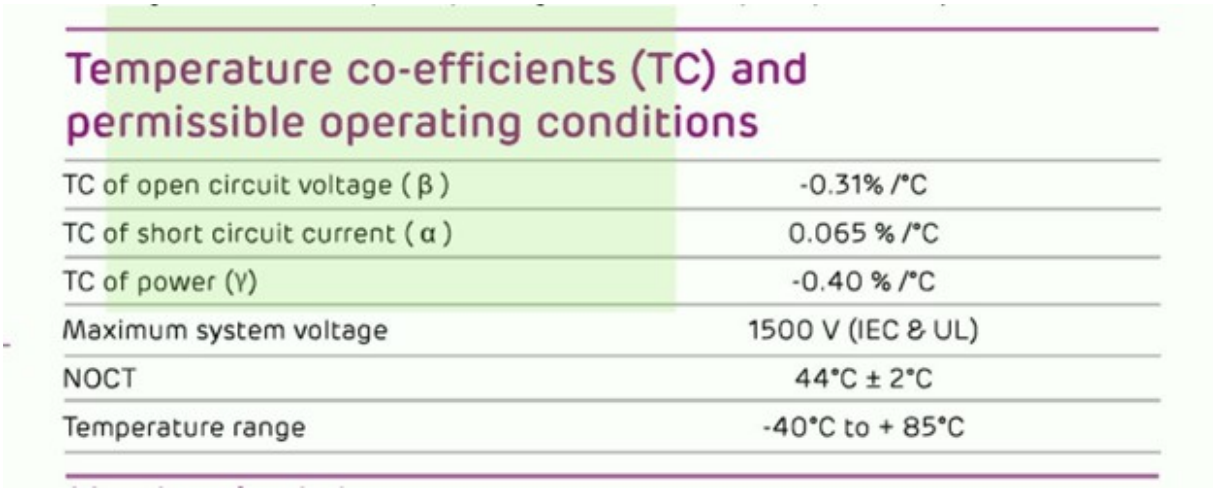


Figure 35: Augmentation of Figure 33 with a brightness factor of 0.80, channel shift of [-1 2 -3], and a random partial mask.

Figure 35 was created by applying augmentation techniques such as color adjustments and random partial masking on Figure 33. Initially, a random brightness factor of 0.80 was applied, followed a RGB channel shift of [-1 2 -3]. This mimics color variations and improves the model's robustness to varying color schemes. Finally, random masks with varying opacities were introduced to occlude parts of the image to help the model infer table structures even when parts of the table are missing.

These techniques helped the model generalize better and improve robustness to noise. It also helped address class imbalance issues by creating additional samples of underrepresented elements.

4.1.4 Data Splitting Strategy

A well-defined Data Splitting strategy offers reliable model training and unbiased evaluation. Accordingly, the augmented PV module datasets were then divided into training, validation, and test subsets in an 80:10:10 ratio. The training set is used to train the model's parameters and allows the Table Transformer (TATR) model to learn structural patterns. This Data Splitting strategy resulted in 169 PDF images for training the Table Detection model and 545 table images for training the Table Structure Recognition model respectively. The validation set was used for hyperparameter tuning and for implementing the early stopping criteria to prevent overfitting.

Finally, the test set was used in the evaluation stage to assess model's generalization capabilities on unseen data and provide an unbiased estimate of the model's performance.

4.2 Data Preprocessing

The Photovoltaic (PV) module datasheets are often distributed in PDF format due to its cross-platform compatibility, uniform appearance across devices, and ease of distribution. However, since PDF's are designed for ease of human understanding rather than structured data processing, extracting this data can be quite challenging. PDF documents have multiple layers that contain different forms of data such as textual, image, and vector-based information. Text-based PDFs contain structured textual content that can be parsed by extraction tools like Camelot [15] and Tabula [30] as observed in the Lightning-Table pipeline described in Section 2.3, whereas image-based PDF's store information as images and therefore require Optical Character Recognition (OCR). Both these formats require different pre-processing steps to standardize the input for further analysis. However, converting all PDF files into images creates a consistent approach for processing both text-based and image-based files, enabling a uniform pipeline for the extraction of structured data from all types of documents. Figure 36 shows the block diagram with the key stages in the pre-processing approach.

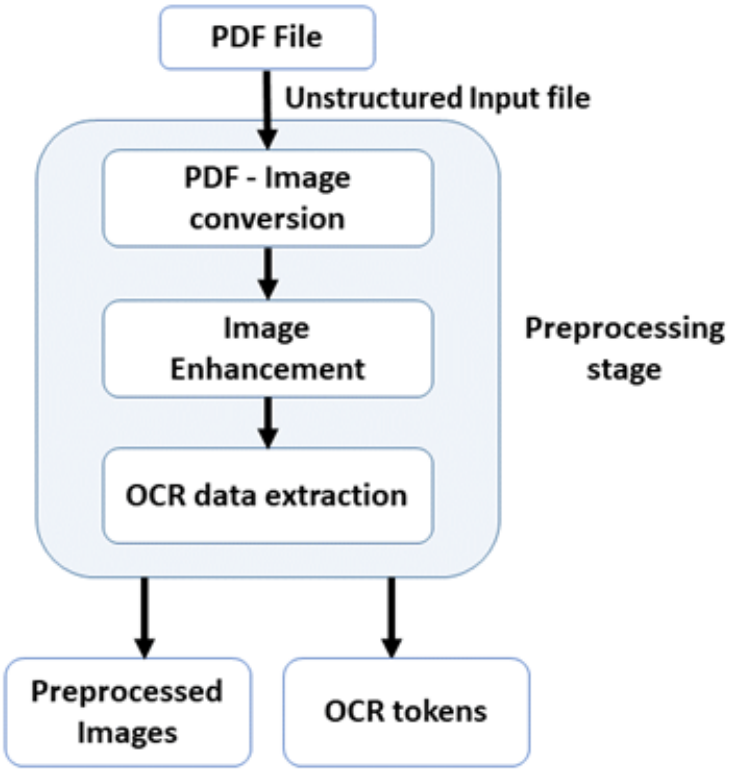


Figure 36: Pre-Processing stage of the propsed Data Extraction Pipeline.

4.2.1 Converting PDF Documents to Images

PDF documents were efficiently converted into PIL (Python Imaging Library) image objects using the Python library pdf2image [42]. The package can also handle multipage PDF files by rendering an image for each page of the input document, as demonstrated in Figure 37. This image extraction technique preserves the layout, text, graphical elements, and embedded tables as they appear in the document. It also tags the rendered images with the page number to maintain page order and ensure that the contextual relationships between tables are preserved to facilitate structured data extraction. This standardization ensures that all pages enter the pipeline in a compatible format thereby, enhancing the reliability and efficiency of subsequent tasks.

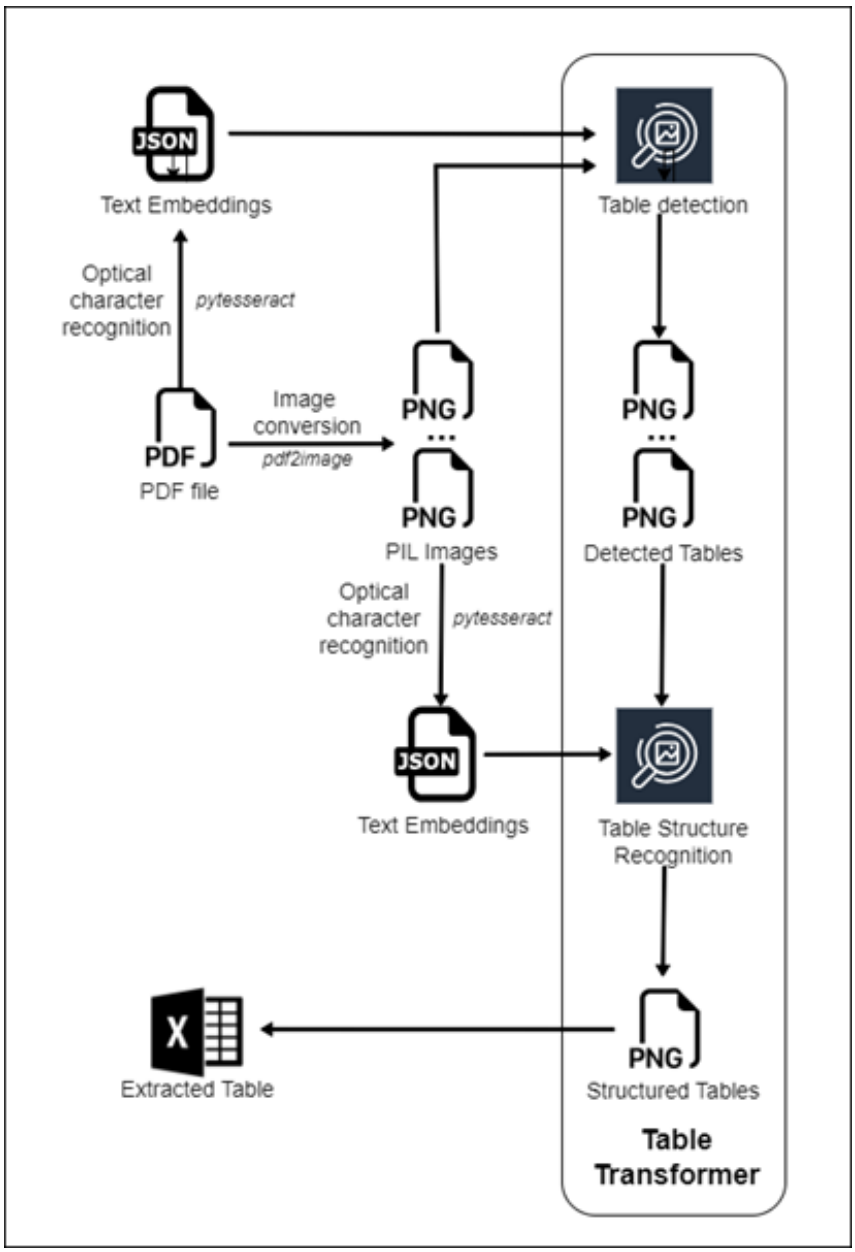


Figure 37: Table extraction using Table Transformer (TATR).

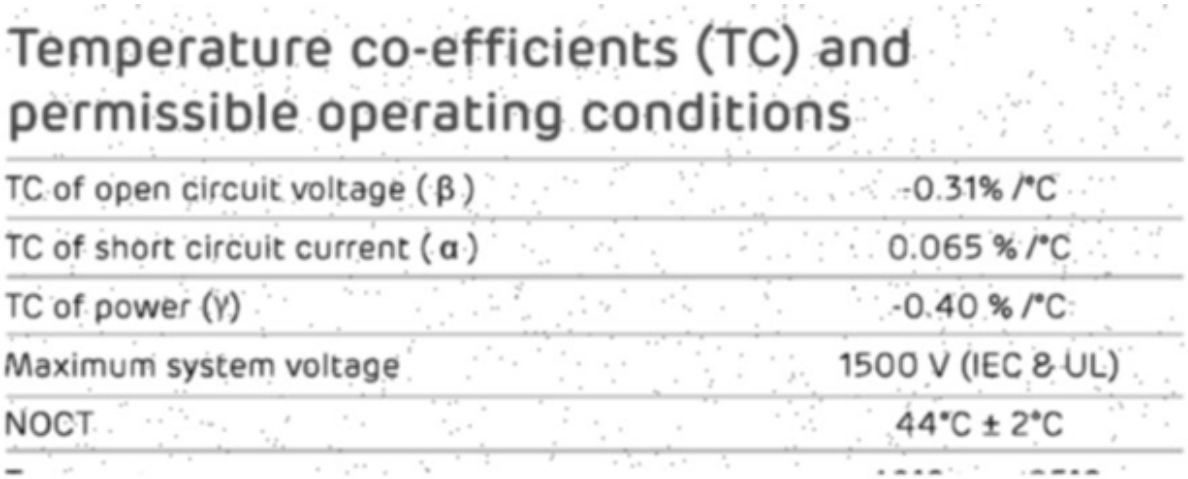
4.2.2 Image Pre-Processing

Although converting PDF files to images ensures standardization and makes document processing easier, it also introduces certain challenges. Managing the resolution of the extracted image is a major challenge as high-resolution images can be computationally expensive in terms of memory and processing requirements. Whereas, low-resolution can result in blurry images and subsequently hinder the performance of the extraction pipeline. Low-resolution images often lead to unrecognizable or misclassified text by the OCR tool when the font size is too small and also causes the detection model to fail when the width of table lines is less.

Therefore image enhancement techniques were employed to preserve the fine details in the document, especially within table cells since it contains critical numerical data regarding PV module performance essential for further analysis. These techniques minimize the challenges posed by noisy and low-resolution images. Each of these techniques were carefully selected to address specific challenges associated with processing technical documents such as PV module datasheets and are as follows:

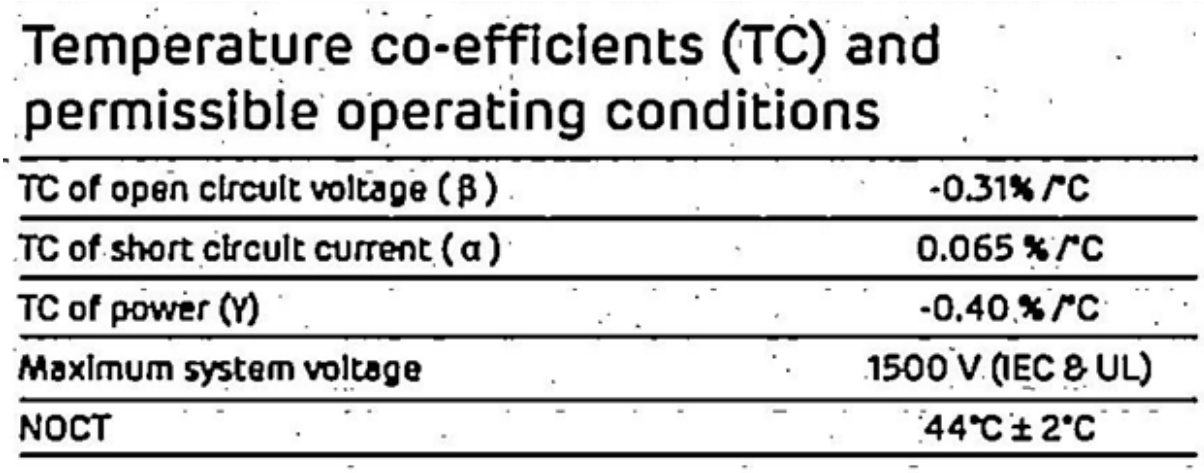
- **Grayscale Conversion :** The RGB images were converted to grayscale images since the removal of color simplifies the image and emphasizes the model's attention on the textual and tabular elements. This step decreases computational complexity and enhances the model's ability to focus on relevant structural features.
- **Noise Reduction :** Bilateral filtering smooths out irregularities resulting from scanning artifacts or image compression, and it also ensures that the text and table lines are sharper by reducing the noise in the images.
- **Thresholding (Binarization) :** Thresholding takes grayscale images as input and converts them into binary images, thereby separating the foreground (text and table lines) from the background. Otsu's thresholding method was used to adaptively determine the threshold value to have a consistent binarization across various PDF color schemes.
- **Morphological Opening :** It consists of an erosion operation followed by dilation for effectively removing small noise and preserving key features. This technique clears isolated scanning artifacts and enhances the clarity of the text.
- **Sharpening :** Sharpening filters enhance the clarity of table lines and text. This step consists of a convolutional kernel that sharpens edges, making cell boundaries and row/column separators more distinct. This addresses blurriness in scanned documents or low-resolution images.

- **Resolution Adjustment :** To resolve issues with small fonts or unclear text, the images are upscaled to increase their resolution. Scaling factors were applied to improve image resolution without loss of quality. For high-quality resampling, LANCZOS technique was used to preserve detail in the upscaled image while avoiding artifacts. This step ensures that even small fonts or fine table lines remain legible by OCR.
- **Padding Adjustment :** Adequate padding is added to the extracted images on all sides to prevent any important details near the edges from being cropped out. This will ensure that the entire table content is included in the analysis and minimizes the chances of data loss.



Temperature co-efficients (TC) and permissible operating conditions	
TC of open circuit voltage (β)	-0.31% /°C
TC of short circuit current (α)	0.065 % /°C
TC of power (γ)	-0.40 % /°C
Maximum system voltage	1500 V (IEC & UL)
NOCT	44°C ± 2°C

Figure 38: Enhancement of Figure 34 by applying grayscale conversion and a bilateral filter to improve text clarity.



Temperature co-efficients (TC) and permissible operating conditions	
TC of open circuit voltage (β)	-0.31% /°C
TC of short circuit current (α)	0.065 % /°C
TC of power (γ)	-0.40 % /°C
Maximum system voltage	1500 V (IEC & UL)
NOCT	44°C ± 2°C

Figure 39: Further enhancement of Figure 38 using adaptive thresholding, morphological opening, and sharpening to enhance text readability.

Figure 38 was obtained by applying image enhancement techniques such as grayscale conversion and filtering to Figure 34. The Bilateral filter reduces noise while preserving edges and is better

than a Gaussian blur for text. Advanced Image Enhancement techniques such as thresholding, morphological opening, and sharpening were applied on Figure 38 to generate Figure 39. The adaptive thresholding technique was used to binarize the image based on local pixel neighborhoods rather than a global threshold. This was followed by morphological operations to remove scanning artifacts and isolated noise. Lastly, a sharpening filter was applied to further emphasize the edges to obtain clearer and more legible text

These image preprocessing techniques enhance the visual quality of the extracted images and help improve the performance of OCR engines and object detection algorithms, thus ensuring more accurate data extraction.

4.2.3 Data Extraction using OCR

Optical Character Recognition deals with converting textual content present in images into machine-readable data. Tesseract, an open-source OCR engine, was used here due to its robust text recognition capabilities and compatibility [32]. The images enhanced in Section 4.2.2 were passed as input to the OCR engine to extract textual content along with their spatial positions, represented as bounding boxes.

Initially, Tesseract processes the input images to detect text at the word level [32]. The OCR engine then segments text into blocks, lines, and words thus, giving a fine-grained representation of the content. Each word has its own bounding box coordinates, line number, and block number, allowing for detailed spatial analysis of the extracted text.

The bounding box defines the spatial coordinates of the text in an image to help align the extracted data with its visual appearance in the datasheet. In addition, the detected text was cleaned to remove extra and non-essential content by blacklisting certain irrelevant characters, such as monetary or trademark symbols (‘€’, ‘£’, ‘©’, ‘™’, etc.) and greek alphabets (‘α’, ‘β’, ‘γ’, ‘δ’, etc.). This process reduced misinterpretations by filtering out irrelevant characters and hence increased the accuracy and reliability of the data extraction pipeline. Finally, metadata, like line and block numbers are added to maintain the logical structure of the document, providing a hierarchical organization of words and phrases. This structured representation of the document is then stored in a JSON file, which easily allows for downstream processing. The automated OCR processing makes this pipeline efficient at handling large datasets and also suitable for batch processing of multiple datasheets.

Tesseract OCR was used to extract the token from Figure 39 [32]. Figure 40-Top illustrates the original image with detected bounding boxes and Figure 40-Bottom depicts the structured visual representation of the extracted tokens and their corresponding bounding boxes. The OCR extraction quality is fairly accurate in capturing text and spatial positioning for this example. From Figure 40, it can be observed that the blacklisted symbols, such as '(α)', '(β)', and '(γ)', were misidentified as '(a)', '(8)', and '(Y)', respectively, while the symbol '&' was not detected at all. Additionally, minor recognition errors were observed in some words, such as *"clrcult"* instead of *"circuit"*. This highlights the drawbacks of using OCR for data extraction and the need for efficient image pre-processing techniques and text post-processing to ensure data accuracy.

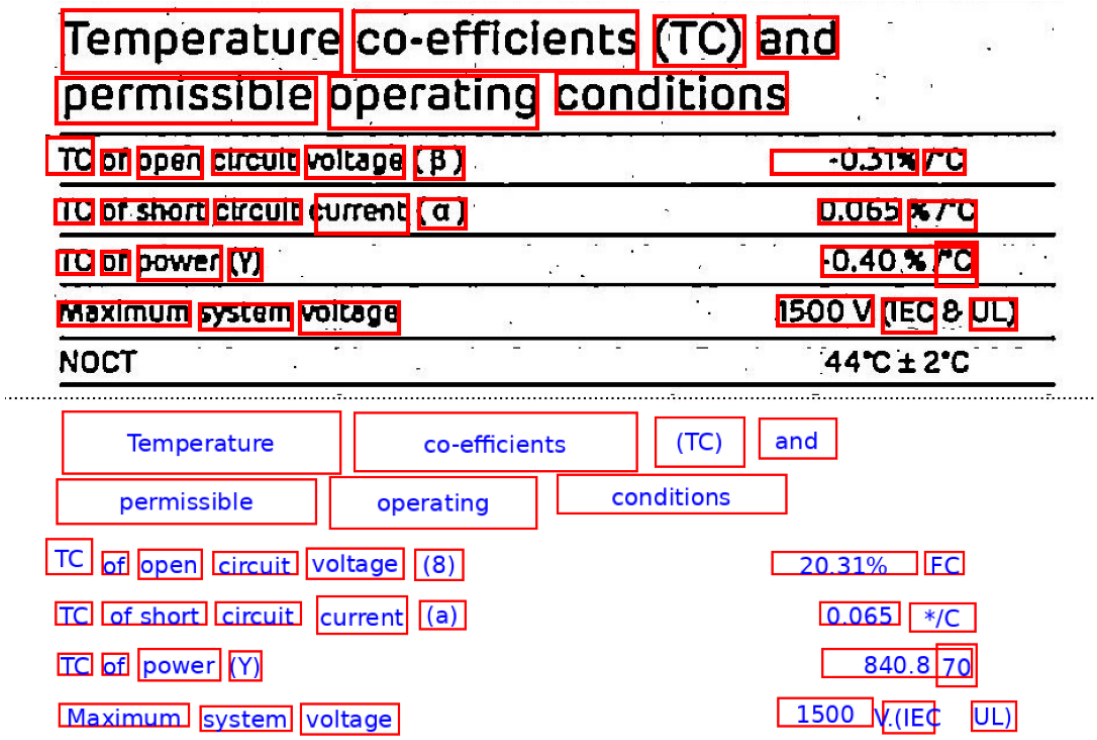


Figure 40: OCR tokens extracted from Figure 39 using Tesseract OCR engine. Top: Original image with detected bounding boxes. Bottom: Spatial representation of extracted text tokens.

4.3 Table Detection

The preprocessed PDF images are used as input to detect tables and identify their location to ensure that only the regions of interest are processed in the subsequent stages and establish computational efficiency. This section outlines the methodology and techniques used to locate tables in PV module datasheets. Figure 41 shows the block diagram of the key processes in the Table Detection (TD) stage.

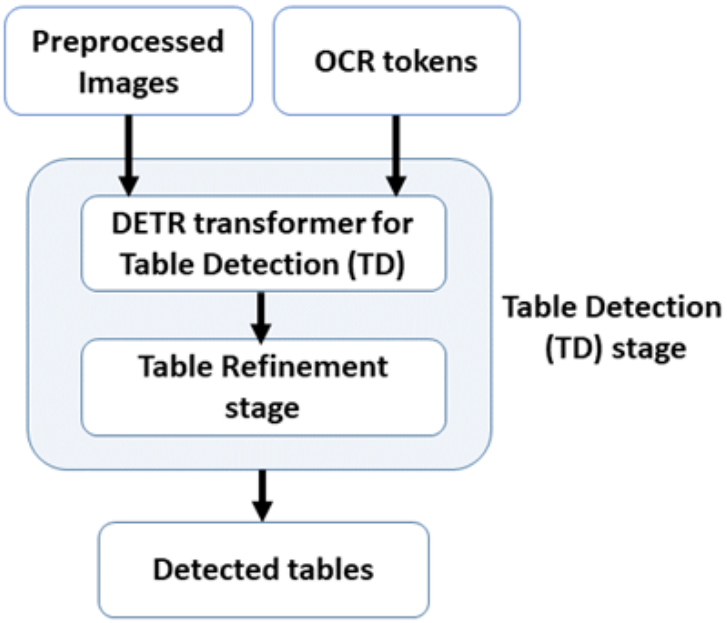


Figure 41: Overview of Table Detection approach.

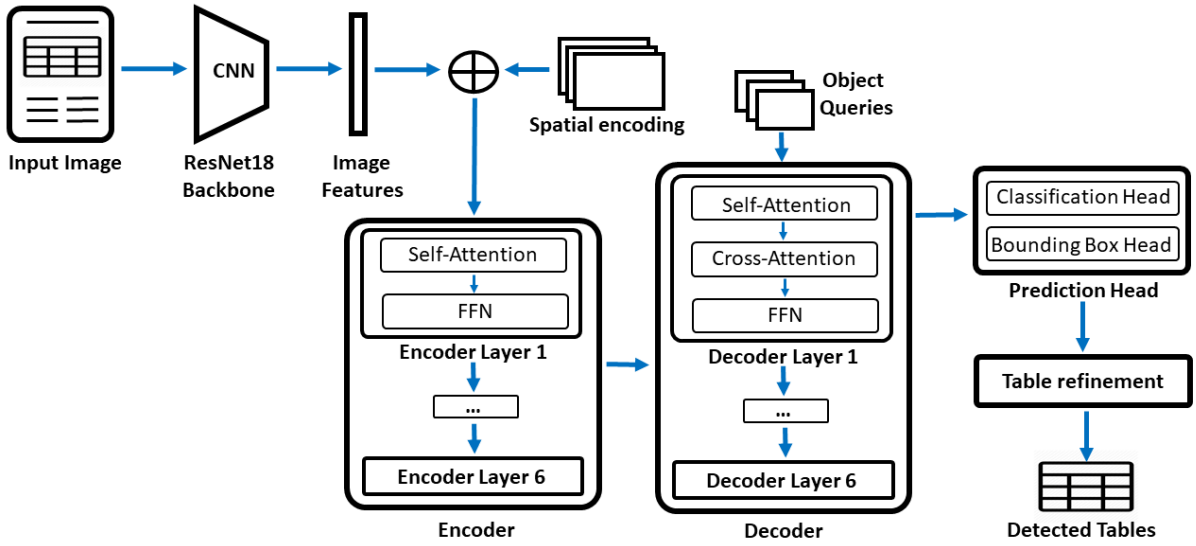


Figure 42: Table Detection architecture diagram.

4.3.1 Methodology

A Detection Transformer (DETR) model discussed in Section 3.3.1 was initially trained on the PubTables-1M dataset and later fine-tuned on the custom PV module training dataset for Table Detection (TD) task created in Section 4.1.4 to identify tables present in the PV module datasheets. Figure 42 depicts the Table Detection architecture diagram and the TD model training is covered in detail in Section 5.2.

At inference, the PDF images extracted using the library pdf2image and enhanced using the techniques described in Section 4.2.2 are provided as inputs to the DETR model along with the corresponding OCR tokens[42]. This multi-modal input ensures accurate table detection in documents such as technical datasheets where tables are present alongside other detailed information, graphs, and diagrams. The table detection process was further broken down into three key stages.

- **Image Transformation for Model Input :** The enhanced images are first transformed by resizing, normalizing, and converting them into tensors. These transformations ensure that the model receives standardized inputs.
- **Model Inference and Output Processing :** The transformed images are then fed into the table detection model, which generates predictions in the following format:
  - **Bounding Boxes :** These are the spatial coordinates that indicate the locations of the detected tables. In case of rotated tables, the prediction head has an additional output channel to estimate the rotation angle. These normalized values are mapped to a full range of possible rotation angles between 0° to 180°.
  - **Confidence Scores :** They provide the confidence levels associated with the detected table. This indicates the likelihood of the detected object.
  - **Class Labels :** These labels identify whether the detected object is a table, a rotated table, or a non-table region.
- **Label Assignment :** The model outputs are processed to assign class labels and filter detections based on the confidence thresholds. Only detections with confidence scores above predefined thresholds were retained for further analysis to ensure reliability and accuracy in identifying tables.

4.3.2 Refining Detected Tables

Once the tables are detected by the DETR model, they are cropped from the original image, based on the generated bounding boxes. To enhance the quality of the tables detected and better visualize the detection results, the following refinement techniques were applied:

- **Padding and Rescaling:** A padding margin was added to these detected table regions to make sure that no table content at the edges was omitted. The cropped images are further rescaled to improve resolution, especially for smaller tables or tables with densely packed content.

- **Image Quality Enhancement:** This step involves sharpening the cropped images and enhancing contrast using the techniques described in Section 4.2.2 to create sharper images of table lines, headers, and text. These enhancements improve the performance of the Table Structure Recognition model and OCR extraction accuracy.
- **Handling Rotated Tables:** If a detected table is classified as a *'rotated table'*, a standard image transformation such as an affine rotation by the negative of the predicted angle is applied to the cropped image to correct its orientation. This maintains consistency with downstream tasks.

For each detected table in the PDF image, the final output of the table detection process includes the cropped table images and associated spatial metadata. The outcomes of this step are shown in Section 5.2.3. This step will ensure that the extracted tables are well-defined, correctly oriented, and ready for further processing.

4.4 Table Structure Recognition

Following the Table Detection stage, the extracted tables, along with the OCR data are passed onto another DETR transformer model for Table Structure Recognition (TSR). This process involves recognizing internal table components and transforming the detected table into a structured machine-readable format. Given the complexity of tables in PV module datasheets, TSR must account for irregularities, such as merged cells, dual-axis tables, and implicit boundaries in the tables. This section outlines the methodology and techniques applied to recognize the structure of tables in PV module datasheets. Figure 43 shows the block diagram of the key processes in the Table Structure Recognition stage.

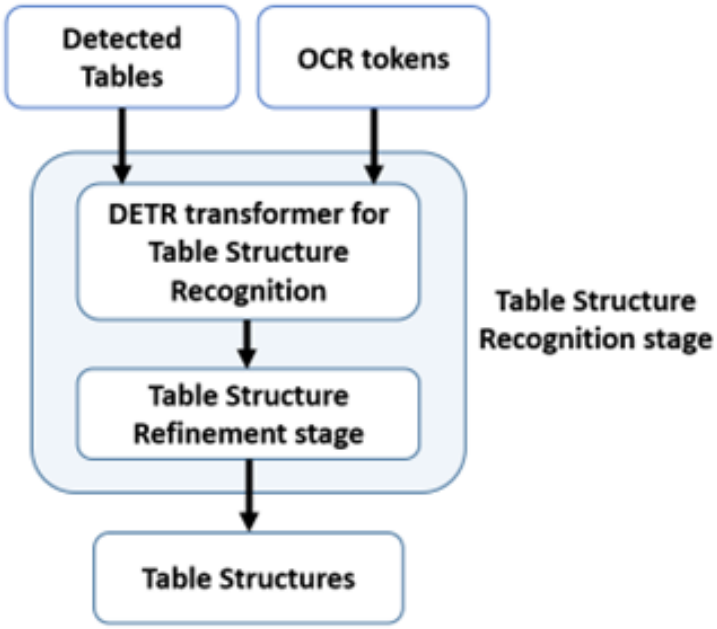


Figure 43: Overview of Table Structure Recognition approach.

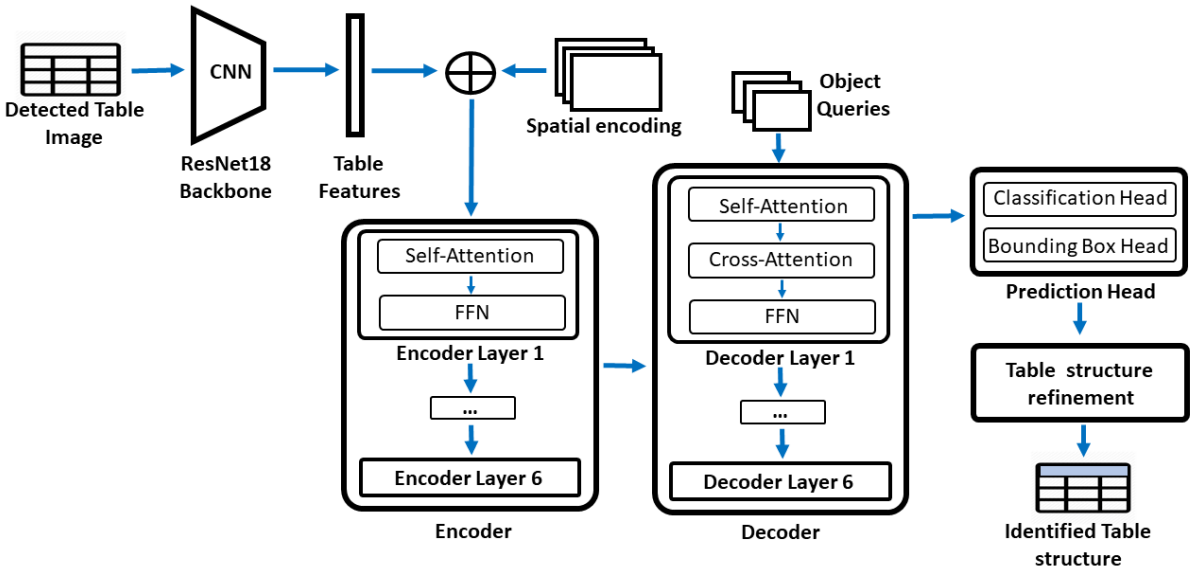


Figure 44: Table Structure Recognition architecture diagram.

4.4.1 Methodology

For TSR tasks, a DETR model discussed in Section 3.3.1, was initially trained on the PubTables-1M and FinTabNet.c datasets which primarily consists of vertical tables that are characterized by columns representing categories and corresponding rows containing values. In contrast to the PubTables-1M training dataset, the PV module datasheets consist of horizontal tables where rows represent categories and the columns contain corresponding values, as well as dual-axis

tables that integrate both layouts. The existing object classes mentioned in Section 3.5 could not accurately represent these table orientations and led to some limitations in recognizing the structure and processing of such tables. To efficiently extract data from these tables, additional structural elements were introduced and the DETR model for TSR task was extended to include the following object classes:

- **Table Name :** This parameter simplifies the Table Classification process in the post-processing stage by only passing the word tokens from the *'table name'* parameter rather than the entire tabular content as input to the Naive Bayes classifier. This process will be discussed in detail in Section 4.7.1. This further reduces the data volume and enhances the computational efficiency of the Naive Bayes classifier.
- **Table Row Header and Table Projected Column Header :** These parameters capture the structure of horizontal and dual-axis tables. They ensure that the model can distinguish between data cells and headers arranged along the column and a table can have multiple row headers.

These additional classes enabled the model to better understand and handle diverse table orientations in PV module datasheets. The TSR model was then fine-tuned on the custom PV module training dataset created in Section 4.1.4 using the updated canonicalization algorithm presented in Section 4.4.3 to recognize the structure of complex tables. The model training and fine-tuning are covered in detail in Section 5.3 and Figure 44 presents the TSR architecture diagram.

At inference, the tables detected in Section 4.3 were provided as input to the OCR engine to generate text embeddings. These embeddings along with the detected tables were passed as input to the trained DETR model for TSR. This multi-modal approach enables a deeper understanding of table structures and ensures that the table elements along with their bounding boxes and associated confidence scores are accurately recognized.

4.4.2 Table Structure Refinement

To ensure consistency and accuracy, the table's structural components recognized in the previous section undergo refinement as follows:

- **Non-Maximum Suppression (NMS) :** NMS removes redundant or overlapping detections by prioritizing detections with the highest confidence scores. This process resolves the containment overlap issue where one detected object partially/fully overlaps another object

and ensures that the detected table components, such as rows, columns, headers, etc., are uniquely and accurately represented.

- **Sorting and Alignment :** The detected rows and columns are spatially sorted to ensure logical ordering. The rows are sorted from top to bottom and the columns are sorted from left to right. This aligns with human-readable conventions and makes downstream processing tasks easier.
- **Handling Table Headers :** Headers, spanning cells, and projected headers were identified and refined following Wang's model [39]. For tables with both row and column headers, measures were taken to ensure proper association of headers with their corresponding cells.

The final output of the TSR stage is the set of original cropped table images, with all its recognized cells defined by their bounding boxes, content, predicted class label and associated spatial metadata. The outcomes of this step are shown in Section 5.3.3. Furthermore, tokens for every individual text element are extracted, along with their spatial coordinates, to allow for precise alignment with table structures. This step will ensure that the extracted tables are well-defined, logically ordered, and ready for further processing.

4.4.3 Canonicalization for Vertical and Dual-Axis Tables

Traditional canonicalization algorithm described in Section 3.5.2 fails to address the intricacies of processing horizontal and dual-axis tables which require further refinements in order to accurately represent and extract data. Hence, the following improvements were included:

- **Horizontal Table Adjustments :** The algorithm ensures that row headers are appropriately extended and that they span the entire height of the table. Multi-level row headers are treated by hierarchically aligning the nodes and merging blank cells that are in the same column.
- **Dual-Axis Table Refinements :** A dual-axis table must be aligned along both axes simultaneously. The algorithm detects overlapping headers and resolves the conflicts by prioritizing logical relationships over physical boundaries.

Special Cases for Complex Structures:

- **Blank Stub Headers :** Tables with blank stub headers (blank cells in the top-left corner of the table) do not consider these cells as part of the logical table structure.
- **Nested Headers :** Multi-level headers with nested categories are refined to ensure that all child nodes align with their parent.

4.5 Table Orientation Detection

Table Orientation Detection (TOD) process determines the dominant axis of a detected table. In order to properly recognize the structural elements, perform canonicalization, and extract data from tables, it is essential to know the table's orientation.

4.5.1 Methodology

Table Orientation Detection starts by scanning along both horizontal and vertical axes, extracting numerical patterns while avoiding other non-relevant text, and computing the variances along both axes. Variance measures the degree of variability of data distribution along each axis. The axis with minimum variation is marked as the dominant axis, under the assumption that values of the same category are typically aligned along the primary axis.

However, to capture dual-axis tables, this detected orientation is then verified by checking for the presence of a row header, column header, or both of these headers in the recognized table output of the TSR model. Tables with negligible differences in the variances defined by a set threshold or tables containing both row and column headers are considered to be dual-axis tables.

4.6 Tabular Data Extraction

The Tabular Data Extraction (TDE) stage extracts the tables from PV module datasheets and ensures that the extracted data is correctly transformed into universally compatible formats while maintaining the structural and logical integrity of the tables. This section outlines the detailed methodology and significance of the table extraction step.

4.6.1 Methodology

The rows, columns, spanning cells, and headers identified and refined at the TSR stage in Section 4.4 were provided as input. Initially, row and column dilation adjusts the bounding boxes to ensure that the partially overlapping cells are correctly aligned within the table structure. In the case of merged cells (cells that span multiple rows or columns), the bounding boxes of those rows or columns are expanded to fully include the merged cell and align it with the table's structure. This process ensures that the rows and columns are of uniform size and shape, improving alignment and preserving the logical organization of the table for further processing.

Text extraction combines the word tokens and corresponding metadata detected by the OCR engine into complete strings for each table cell. The text is then logically sorted to match the table's structure. This process maps each cell to its row and column using bounding box coordinates, handling issues like merged cells, varying sizes, and overlaps. This cell alignment process ensures that the extracted content accurately reflects the cell's position within the table.

While Table Transformer effectively identifies the table's structure, additional processing is required to fill the extracted data in the final Excel document output with a simple table structure. Data from merged or spanning cells is extrapolated into individual cells based on the updated canonicalization algorithm described in Section 4.4.3. This process involves analyzing the table structures, identifying the merged cells, and duplicating their content across the corresponding rows and columns.

The final output of this stage is a set of well-structured CSV or Excel files representing the extracted tabular content, making it easier to utilize these results in existing analytical tools and workflows. Each extracted table is stored in its own worksheet within a single Excel file named after the source PDF document. This systematic organization enables efficient analysis and ensures that the data remains traceable to its original input document. The automation of these processes allows the pipeline to handle large volumes of datasheets efficiently, making it suitable for industrial-scale data extraction tasks.

4.7 Postprocessing Tabular Data

In this stage, the key information present in the PV module datasheets are identified, validated, and extracted for further analysis from the structured Excel documents generated in Section 4.6.1. The post-processing stage employs Machine Learning techniques such as the Naive Bayes classifier for classifying the extracted tables, and rule-based methods such as regex pattern matching and LLMs to address inconsistencies and variability in tabular data to produce structured outputs. This section outlines the methodology and techniques applied for post-processing and extracting relevant information. Figure 45 depicts the block diagram of the key components in the data post-processing stage.

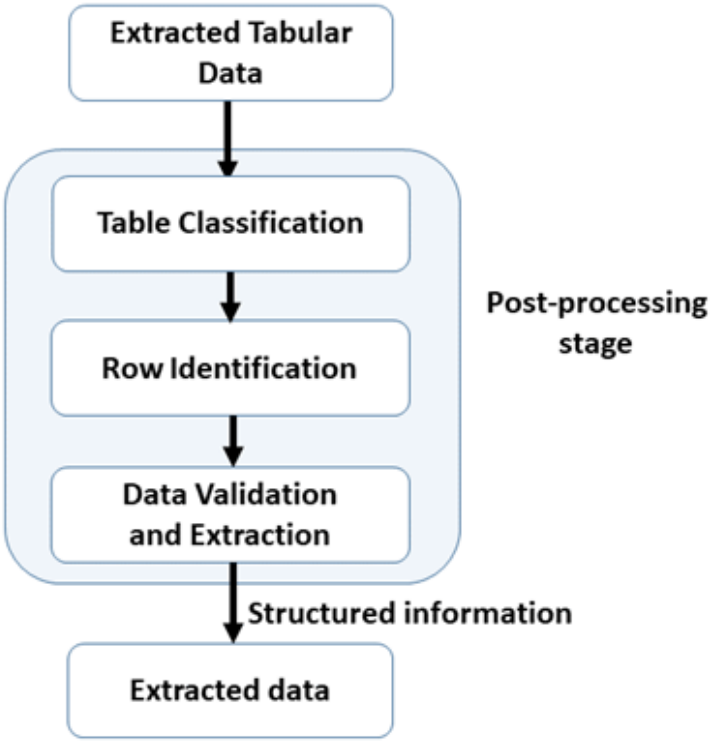


Figure 45: Overview of the Post-processing stage.

4.7.1 Table Classification

This step involves correctly identifying the class or type of each table (worksheet) present in the extracted excel output. Every table type has its own set of rules for processing and correctly extracting values. Since the structure and naming conventions followed by different manufacturers are diverse, identification based on table titles alone is usually unreliable. Therefore, text classification methods described in Section 3.4 were employed.

The tables are processed to extract keywords from their rows and columns, which are then vectorized using the TF-IDF (Term Frequency-Inverse Document Frequency) word embedding technique as described in Section 3.4.1. If the corresponding table has an *'table name'* parameter, then only the words contained within this parameter are processed. Word vectors capture the contextual relationships between terms, even when documents contain OCR errors or inconsistencies in language. Word vectorization techniques like Word2Vec enable the model to identify similar terms, therefore making it resilient to variation in terminology [43]. For example, *'short current'* is often used as an abbreviation for *'short circuit current'*. Vectorization would capture the contextual proximity between these terms, thus improving classification accuracy.

A naive Bayes classifier is then applied to predict the table class based on the vectorized representation of its textual features extracted, as described in Section 3.4.2. This ensures robust classification across various table layouts. Section 5.5 covers the detailed Naive Bayes classifier training process.

4.7.2 Row Header or Column Header Identification

This step involves identifying specific rows in the case of horizontal tables or identifying specific columns in the case of vertical tables that contain the desired information. This task can be challenging due to the diverse naming conventions followed by different PV manufacturers and the organization of information in the headers.

Regular Expression Matching

Once the tables are classified and their orientation is determined, the next task is to locate the rows or columns containing relevant information. This step uses regular expression (regex) pattern matching to identify specific textual or numerical patterns within table cells.

The classified tables from Section 4.7.1 are passed as input to this stage, along with their orientation information from Section 4.5. Regular Expressions are a sequence of characters designed to match specific text patterns and are an efficient tool for locating textual data. A repository of predefined regex patterns was created for relevant values in the PV modules datasheet. In the case of horizontal tables, each cell in the row headers is scanned against these patterns, and in the case of vertical tables, each cell in the column headers is scanned. For the Dual-axis table, both row and column headers are processed to identify the regex pattern. Once a match is found, the corresponding rows or columns are selected. Thus, ensuring that only relevant data is extracted.

Large Language Models

Large Language Models (LLMs) such as GPT-4 can also be leveraged to locate the rows or columns containing relevant information, due to their ability to understand natural language descriptions and identify specified information within tables [44]. The regex patterns often fail when data deviates from expected formats, whereas, the context-aware nature of LLMs provides flexibility and robustness when handling these unstructured or poorly organized data.

A repository of natural language descriptions is maintained for the key values, such as electrical specifications, thermal coefficients, or mechanical dimensions. These descriptions act as prompts for the LLM to locate the required data. Unlike regular expression which requires predefined patterns, these descriptions make LLMs more adaptable to variations in table structures and naming conventions followed by different PV manufacturers. They are also easier to maintain and update when dealing with complex and evolving datasets.

Manual testing was conducted by passing the classified tables from Section 4.7.1 along with the corresponding prompts as inputs to the model by using online interfaces. The model then interprets these prompts to extract the requested data. This provides a qualitative evaluation of the model's ability to handle variations in table structure and table misclassifications. This approach highlighted LLMs' flexibility and robustness compared to regex-based methods, although it did not provide quantitative metrics on real-time performance, scalability, or integration efficiency.

Figure 46 depicts the example of a table classified as an *'Electrical Characteristics at Standard Testing Conditions (STC)'* table. However, this table contains crucial values such as module efficiency, voltage, current, etc. at the NOTC condition as well. The data extracted from this table was passed as input to the LLM along with the corresponding natural language descriptions to extract the *'Module Efficiency'*. The prompt used for this extraction is *'From the electrical characteristics table, extract the module efficiency (in percentage) for Standard Testing Conditions (STC) only. Round the value to two decimal places.'* Figure 47 compares the extraction results obtained from both regular expression matching and LLM approach. While regex, with its rigid patterns, retrieved all data that matched the described format, LLMs on the other hand, understood the contextual relationships in the table and recognized variations in header names. As a result, the LLM provided a higher data extraction accuracy with fewer errors for this example.

Electrical Characteristics STC: AM1.5 1.000W/m2 NOTC: AM1.5 800W/m2 20º 1 m/s Test uncertainty for Pmax +-3%

Module type	TM - 650 M-132 HC		TM - 655M-132 HC		TM - 660 M-132 HC		TM - 665M-132 HC		TM - 670 M-132 HC	
Testing condition	STC	NOTC	STC	NOTC	STC	NOTC	STC	NOTC	STC	NOTC
Maximum Power (Pmax/w)	650	484	655	487	660	491	665	495	670	498
Open Circuit Voltage (Voc/V)	45	42,6	45,2	42,8	45,4	43,0	45,6	43,1	45,8	43,3
Short Circuit Current (Isc/A)	18,39	14,41	18,43	14,43	18,47	14,48	18,51	14,57	18,55	14,58
Voltage at Maximum Power (Vmp/V)	37,6	35,7	37,8	35,9	38	36,05	38,2	36,1	38,4	36,3
Current at maximum Power (Imp/A)	17,29	13,56	17,33	13,57	17,37	13,62	17,41	13,71	17,45	13,72
Module Efficiency (%)	20,90%	15,58%	21,10%	15,68%	21,30%	15,81%	21,40%	15,94%	21,60%	16,03%

Figure 46: A table extracted by the proposed pipeline and classified as 'Electrical Characteristics at Standard Testing Conditions (STC)' table [24].

Efficiency extracted using RE : ["20.90%", "15.58%", "21.10%", "15.68%", "21.30%", "15.81%", "21.40%", "15.94%", "21.60%", "16.03%"]

Efficiency extracted using LLM : ["20.90%", "21.10%", "21.30%", "21.40%", "21.60%"]

Figure 47: Module efficiency data extracted from Figure 46 using Regular expression pattern matching and LLM.

4.7.3 Data Validation and Extraction

The identified rows or columns from Section 4.7.2 that contain key information then underwent data extraction. Regex patterns were used again to locate and extract the required data from the selected cells. Extracted values are then validated against predefined constraints, such as expected ranges or formats, to ensure accuracy. For example, electrical specifications like *Efficiency* have specific percentage ranges. This step ensures that all the extracted values align with domain-specific expectations and prevents faulty data in the final output.

Finally, the extracted structured output is saved in an Excel file. The extracted data was organized to ensure its compatibility with the existing data visualization tools as seen in Figure 48 to gain useful insights from PV module datasheets.

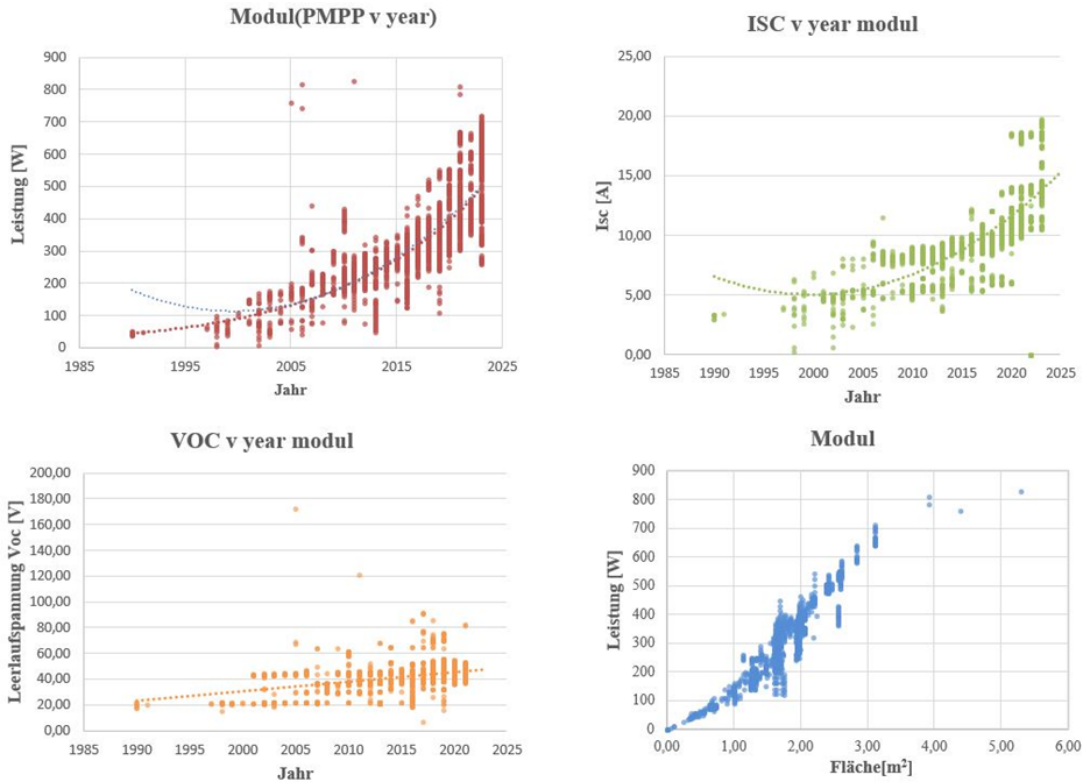


Figure 48: Visualization of extracted tabular data to aid PV research.

5 Experiment and Evaluation

This section presents the design, implementation, and evaluation metrics for assessing the proposed tabular data extraction performance. The experiments were designed to evaluate the model's ability to detect tables in the datasheets, recognize their internal structure, extract the tabular data, and finally, post-process and extract critical PV data. These experiments provide insights into the model's capabilities while measuring the proposed approach's effectiveness and generalizability.

5.1 Evaluation Metrics

This section presents the evaluation metrics to understand the model's performance and provide key insights into areas for improvement. These metrics provide insights into content accuracy, structural fidelity, and spatial precision while also providing a framework for evaluating the model's performance on Table Detection (TD), Table Structure Recognition (TSR), and Tabular Data Extraction (TDE) tasks.

5.1.1 Content Accuracy (Accuracy_Con)

Content Accuracy measures the model's precision in recognizing and extracting text content from table cells. It is calculated by comparing the content of each predicted cell with the corresponding ground truth cell content. For binary classification, the content accuracy is calculated as follows:

$$\text{Accuracy\_Con} = \frac{\text{Number of Correct Cells}}{\text{Total Number of Cells}} \tag{10}$$

Formula 10: Formula for computing the content accuracy for a binary classifications task

Traditionally, content accuracy is a binary measure that considers a match only when the entire cell content is correct, but to handle minor OCR errors or textual variations, a partial matching using Levenshtein distance (string metric for measuring the difference between two sequences) was incorporated with content accuracy to mitigate overly penalizing minor recognition errors [45]. In this case, the content accuracy metric is calculated by summing the partial match scores across

all cells and normalizing them by the total number of cells. This allows for slight OCR-induced variations without entirely discounting near-perfect extractions. The modified content accuracy metric is computed as:

$$\text{Accuracy\_Con} = \frac{\sum_{i=1}^{N} \text{PMS}_i}{N} \tag{11}$$

Formula 11: Formula for computing the content accuracy

where, $\boldsymbol{PMS_i}$ is a *partial match score (Levenshtein distance-based similarity score) for cell i* and $\boldsymbol{N}$ is the *total number of cells.*

This approach provides a more flexible evaluation criterion since the data extracted using Tesseract has a significant amount of OCR-induced errors.

5.1.2 Grid Table Similarity (GriTS)

The Grid Table Similarity (GriTS) metric introduced by Smock et al. [46] was designed to evaluate structured data extraction from tables by assessing the model's accuracy in capturing table layout, content location, and cell relationships. The metric comprises three variants:

(a) Topological GriTS (GriTS_Top)

Topological GriTS measures the model's ability to capture cell adjacency and relational structure within a table. The metric is calculated by comparing the predicted adjacency matrix, which indicates the relationships between cells, with the ground-truth adjacency matrix.

$$\text{GriTS\_Top} = \frac{\text{Number of Correct Relationships}}{\text{Total Relationships in Ground Truth}} \tag{12}$$

Formula 12: Formula for computing the Topological GriTS

This metric highlights the model's capability to understand table structure, which is critical for applications where table context and logical arrangement affect data interpretation and accurate data extraction, particularly in technical and multi-dimensional tables.

(b) Content GriTS (GriTS_Con)

Content GriTS focuses on the model's accuracy in associating content with the correct cell structure. This metric is calculated by comparing the content-labeled grid of cells between the ground truth and the prediction.

$$\text{GriTS\_Con} = \frac{\text{Number of Correctly Matched Content Cells}}{\text{Total Number of Cells in Ground Truth}} \tag{13}$$

Formula 13: Formula for computing the Content GriTS

This is similar to Accuracy_Con but it also includes the structural arrangement and spatial relationships of cells within a table. This metric ensures that the extracted content is placed in the correct cell locations.

(c) Location GriTS (GriTS_Loc)

Location GriTS assesses the accuracy of cell positioning within the table. The metric is calculated by comparing the coordinates of each predicted cell with its ground truth location.

$$\text{GriTS\_Loc} = \frac{\text{Number of Correct Cell Locations}}{\text{Total Number of Cells}} \tag{14}$$

Formula 14: Formula for computing the Location GriTS

This metric ensures that the extracted data maintains spatial integrity, which is crucial for representing the logical structure of tables.

These three GriTS variants provide insight into structural fidelity, spatial consistency, and content precision necessary for robust table extraction. GriTS, introduced by Smock et al. [46], assesses cell topology recognition (GriTSTop), cell content recognition (GriTS_Con), and cell location recognition (GriTSLoc) using the following overall formula:

$$\text{GriTS}_f(\mathbf{A}, \mathbf{B}) = \frac{2 \cdot \sum_{i,j} f(\tilde{\mathbf{A}}_{i,j}, \tilde{\mathbf{B}}_{i,j})}{|\mathbf{A}| + |\mathbf{B}|}. \tag{15}$$

Formula 15: Formula for computing the GriTS score

where, $\boldsymbol{A}$ is the *ground truth matrices of grid cells* and $\boldsymbol{B}$ is the *predicted matrices of grid cells.*

5.1.3 Table COCO Metrics

The Common Objects in Context (COCO) metrics were originally designed for evaluating object detection models but are adapted here for table detection and cell localization [47]. These metrics provide insights into the model's precision and recall performance across various Intersections over Union (IoU) thresholds.

Intersection Over Union

Intersection over Union (IoU) measures the overlap between a predicted and ground truth bounding box and is calculated as the intersection area divided by the union area as seen in Figure 49.

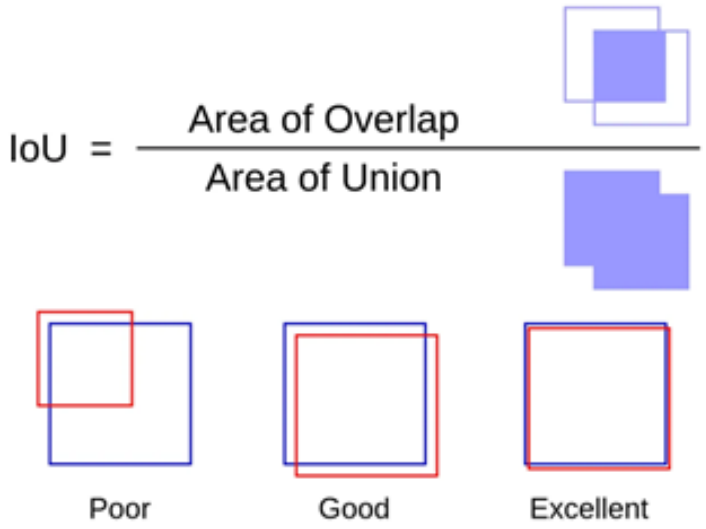


Figure 49: Diagram representing the IoU overlaps and object detection performance [25].

Confusion Matrix

The Confusion Matrix is a table that summarizes the performance of a classification or detection model by comparing predicted labels with actual ground truth labels as shown in Figure 50. This gives rise to:

- **True Positives (TP) :** True Positives are instances where the model correctly identifies an object that matches the ground truth bounding box with sufficient IoU overlap.
- **False Positives (FP) :** False Positives occur when the model incorrectly identifies an object that does not correspond to any ground truth bounding box. It could result from detecting a non-existent object or producing a bounding box with insufficient overlap with the actual cell in the ground truth.
- **False Negatives (FN) :** False Negatives occur when the model fails to correctly identify or localize an object present in the ground truth.
- **True Negatives (TN) :** True Negatives occur when the model correctly identifies no object in a given region, aligning with the ground truth, which also has no object in that area.

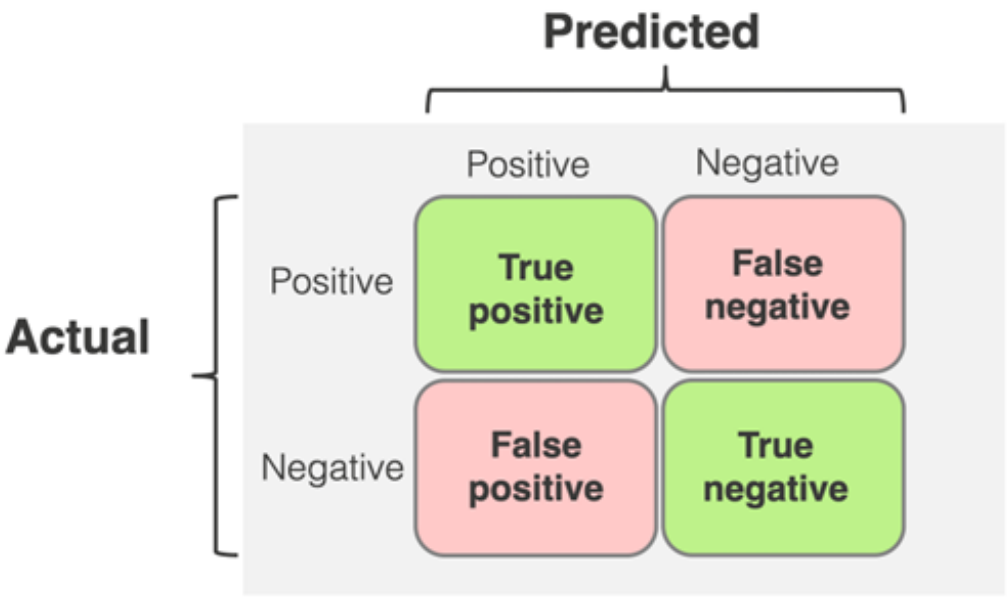


Figure 50: Confusion matrix [26].

The Confusion matrix allows for easy calculation of several key metrics, including Precision, Recall, Accuracy, and F1 Score, which are derived as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{16}$$

Formula 16: Formula for computing the precision

$$\text{Recall} = \frac{TP}{TP + FN} \tag{17}$$

Formula 17: Formula for computing the recall

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{18}$$

Formula 18: Formula for computing the accuracy

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{19}$$

Formula 19: Formula for computing the F1 score

Average Precision (AP)

Average Precision represents precision across various IoU thresholds in specified ranges, offering a comprehensive view of the model's detection performance. In table extraction, AP reflects how well the model detects cells with a wide range of potential overlaps with ground truth.

Average Recall (AR)

Average Recall provides insight into the model's capacity to capture the entire structure of a table, which is crucial for accurate data extraction. An AR score approaching 1.0 indicates a high rate of cell detection, while lower scores suggest cells may be missed during the detection process.

Average Precision at IoU Thresholds of 0.5 and 0.75 (AP50 and AP75)

AP50 and AP75 measure the model's precision based on IoU thresholds. An IoU threshold of 0.5 (AP50) allows for a 50% overlap between predicted and ground truth bounding boxes, while AP75 requires a stricter 75% overlap. The formula for Average Precision (AP) is defined as follows:

$$\text{AP} = \int_0^1 \text{Precision}(r)\, dr \tag{20}$$

Formula 20: Formula for computing the average precision (AP)

where, $\boldsymbol{r}$ represents *recall*.

5.2 Experiment - Table Detection

Section 4.3 discussed the approach for detecting tables from PDF documents in detail. This section will provide the experimental setup and present the evaluation results of this approach. It also includes an analysis of the challenges faced, solutions implemented, and performance metrics achieved during the experimentation.

5.2.1 Experiment Setup

The dataset preparation process, as outlined in Section 4.1.2, describes how the custom solar module data was created and how the data was annotated for table detection. Furthermore, Section 4.1.4 presents how the data was sampled. These steps played a pivotal role in ensuring the success of this experiment.

Pre-Training and Initial Evaluation

Due to the limited size of the annotated training dataset created in Section 4.1.4, the Detection Transformer (DETR) model was initially pre-trained on the PubTables-1M dataset. This dataset consists of 460,589 annotated document pages with tables for the table detection task, including their bounding boxes and text for each word in the tables. The DETR model's performance was evaluated using standard metrics such as AP50, AP75, Average Precision (AP), and Average Recall (AR). When evaluated on the solar module test dataset, the pre-trained model attained an AP50 metric of 50.7%. However, the model showed limitations because the PubTables-1M dataset differs significantly from solar module datasheets. For example, PV module datasheets could have different styles, colors, fonts, and complex backgrounds depending on the manufacturer's branding schemes. The pre-trained model was also designed only to detect table content, whereas capturing table headers is crucial in this pipeline for further table classification. Additionally, PV module datasheets can include customized fonts that are not represented in the PubTables-1M dataset.

Fine-Tuning on the Solar Module Dataset

To address the challenges mentioned above, the pre-trained DETR model was then fine-tuned by adjusting the pre-trained model's weights on the PV module training dataset created in Section 4.1.4. This process involved freezing the ResNet backbone and the early transformer layers of the model so that their weights were not updated during fine-tuning. This process retains the general features learned during pre-training, while selectively updating only the later transformer layers and prediction head observed in Figure 42 to learn the domain-specific patterns and enable the model to capture the table header as well. The model's performance substantially improved, achieving an AP50 of 87.8% on the table detection task.

Furthermore, the OCR data generated in Section 4.2.3 was also given as input to the transformer model in addition to images so that the model can detect tables that lacked proper bounding lines. However, this resulted in a large number of false positives, wherein the model often misclassified well-structured text blocks or graphs as tables. This slightly reduced the model performance resulting in an AP50 score of 87.1%. To mitigate these false positives, the following techniques were employed:

- *Inclusion of Negative Samples :* Non-tabular elements such as graphs and structured text were included in the training dataset to help the model learn the difference between tables and non-table elements.

- *Weighted Loss Functions :* The loss function was adjusted to penalize false positives more severely and enhance the model's ability to handle imbalanced datasets.

Including negative samples also stabilized the training process, and the model achieved an AP50 of 88.7% on the test dataset. The stabilized model was then subjected to Hyperparameter Optimization, where the parameters such as learning rate, batch size, regularization, and weight decay were modified sequentially to derive the optimal configuration and improve model performance, which resulted in an AP50 score of 89.5%. The hyperparameter configuration of the best performing model is as shown in Table 1.

Hyperparameter	**Value**
Learning rate	5×10^{-5}
Batch size	4
Weight decay	1×10^{-4}
LR scheduler	Exponential with Gamma = 0.9
LR drop	1

Table 1: Optimized hyperparameter configuration for the best-performing detection model.

5.2.2 Quantitative Results

The results of various experiments are summarized in the table below:

Model	**AP50**	**AP75**	**AP**	**AR**
Pre-trained (PubTables-1M)	50.7%	14.1%	24.9%	36.8%
Fine-Tuned (Solar Module Dataset)	87.8%	80.5%	66.5%	75.4%
Fine-Tuned with Words	87.1%	81.8%	67.3%	74.3%
Fine-Tuned with Negatives	88.7%	81.1%	66.0%	72.6%
Fine-Tuned with HPO	89.5%	80.9%	68.6%	77.5%

Table 2: Performance comparison of Table Detection models across different training strategies.

The results in Table 2 demonstrate a significant improvement in the TD model's performance through successive refinements. The inclusion of words and negatives had a mixed impact on the AP metrics but they demonstrated robustness in handling complex table structures. The fine-tuned models consistently outperform the pre-trained baseline, with AP75 (a stricter overlap

constraint) increasing from 14.1% to 80.9% on the best-performing model achieved through HPO. The overall precision (AP) and recall (AR) significantly improved to 68.6% and 77.5% respectively. This demonstrates the model's ability to effectively identify tables in densely packed PV module datasheets.

5.2.3 Visualization of Results

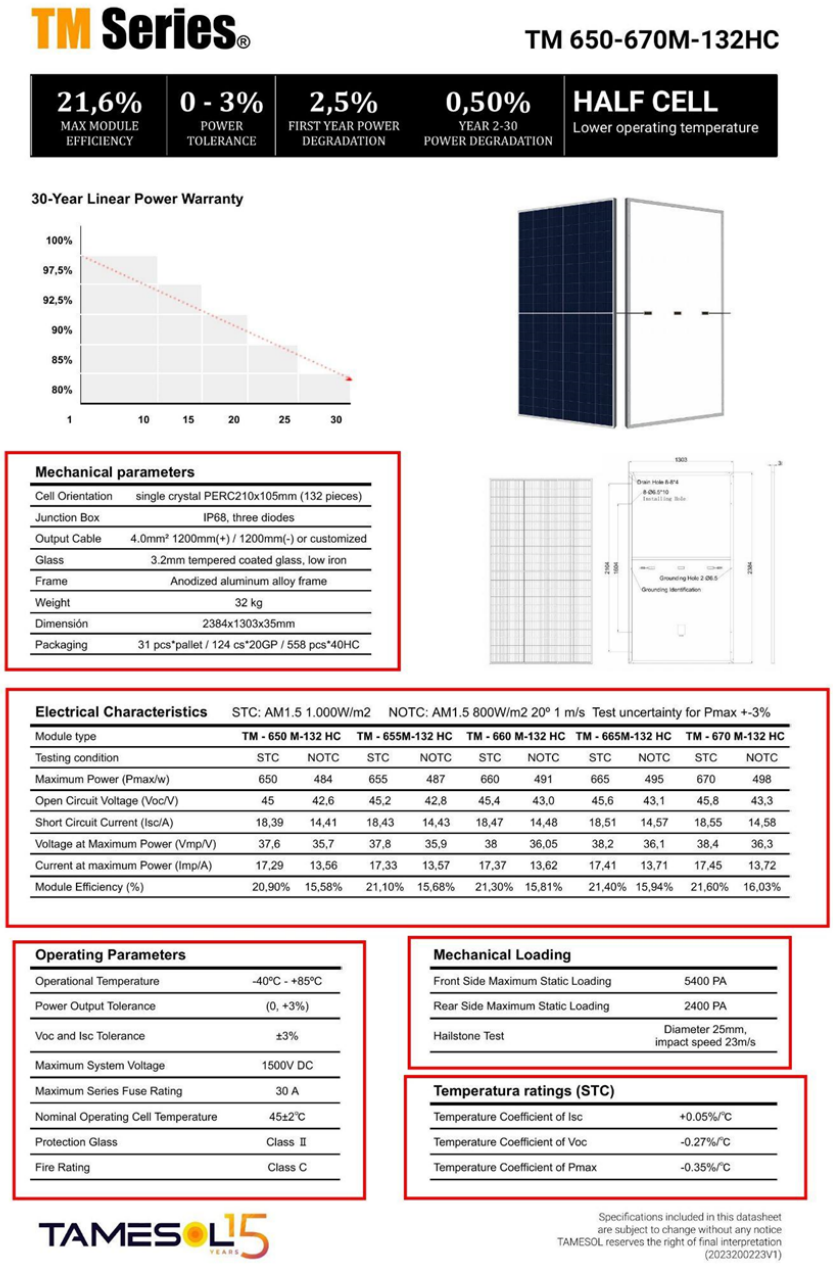


Figure 51: A solar module datasheet with red bounding boxes denoting the tables detected by the TD model.

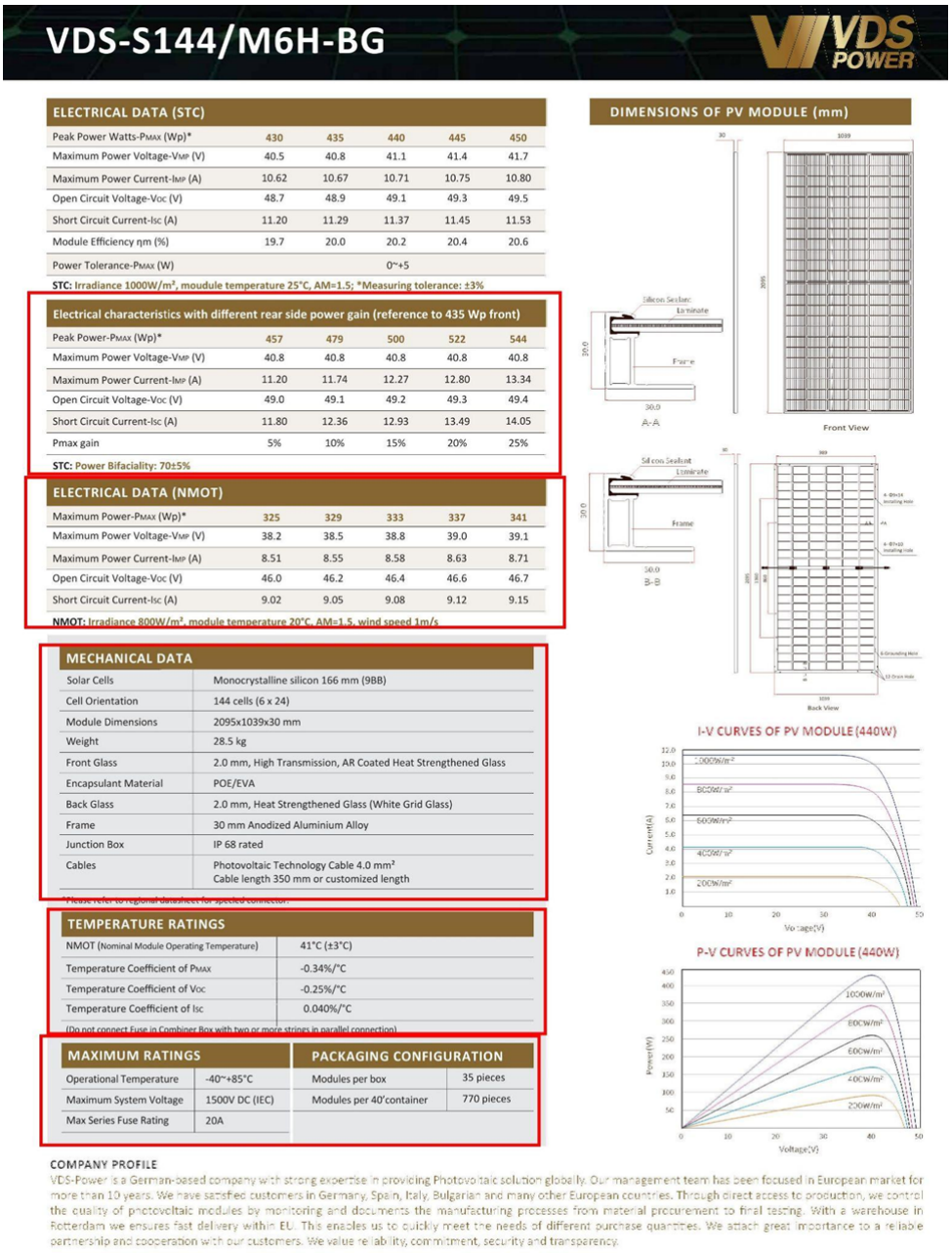


Figure 52: A solar module datasheet with red bounding boxes denoting detected tables. Some tables are missed, while others are incorrectly merged, depicting the drawbacks of the TD model.

Figure 51 and Figure 52 illustrate examples of solar module datasheets, with the red bounding boxes indicating the table detections produced by the best-performing DETR model, and Table 3 presents the detection performance of Figure 51 and Figure 52.

Although the model accurately detected all the tables in Figure 51, the AP75 score remained at 64.2%. This low score results from the inaccuracies in the manual annotation of the ground truth bounding box position or size, which reduces the Intersection over Union metric required for higher thresholds like AP75. This example highlights the sensitivity of the evaluation criteria

used. Additionally, the poor performance of Figure 52 is due to several tables not being detected and others being incorrectly merged.

Image	AP50	AP75	AP	AR
Figure 51	64.2%	64.2%	54.3%	70.0%
Figure 52	38.3%	38.3%	32.4%	50.1%

Table 3: Detection evaluation results of Figure 51 and Figure 52

5.3 Experiment - Table Structure Recognition

This section will provide the experimental setup and present the evaluation results of the approach discussed in Section 4.4. It also includes an analysis of the challenges faced, solutions implemented, and performance metrics achieved during the experimentation.

5.3.1 Experiment setup

Pre-Training and Initial Evaluation

Due to the limited size of the annotated training dataset created in Section 4.1.4, the DETR model was initially pre-trained on the PubTables-1M dataset. This dataset has 947,642 fully annotated tables, including text and bounding boxes for every word for the TSR task. It covers a wide range of table layouts and is a good baseline for transfer learning. When tested on the solar module dataset, the pre-trained model achieved an AP50 score of 10.4%. However, it had major limitations because the PubTables-1M dataset differs significantly from solar module datasheets. Most tables in PubTables-1M are vertical, but solar module datasheets often include horizontal and dual-axis tables. Solar tables can also have complex designs that deviate from Wang's model, which was a key assumption made by TATR [39]. In addition, many tables in solar module datasheets do not have clear rows and columns, making structure recognition more difficult.

Fine-Tuning on the Solar Module Dataset

The pre-trained DETR model was then fine-tuned on the custom PV module dataset created in Section 4.1.4 by freezing the initial layers of the TSR model seen in Figure 44. In this case, the weights of the ResNet backbone and the early transformer layers were not updated during fine-tuning to retain the general features learned during model training on a large dataset. The

weights of the later transformer layers and prediction heads were updated to understand the domain-specific patterns. This approach ensures the model's proficiency in recognizing the unique layouts and complexities of solar datasheets while leveraging the general recognition capabilities of the pre-trained network. Techniques such as Xavier Initialization were leveraged to ensure stable and efficient training. This technique optimizes the variance of input and output connections to mitigate issues like vanishing or exploding gradients. This model resulted in an AP50 metric of 9.1%, highlighting the need for further adaptations to the training dataset and the underlying canonicalization algorithm to refine the recognized table structures. To address these challenges and capture horizontal and dual-axis tables, three new parameters, namely *'table name'*, *'table row header'*, and *'table projected column header'*, were introduced, as specified in Section 4.4.1. The canonicalization algorithm was modified to integrate these additional parameters as mentioned in Section 4.4.3 and enable the model to effectively handle table structures not present in the original pre-training dataset.

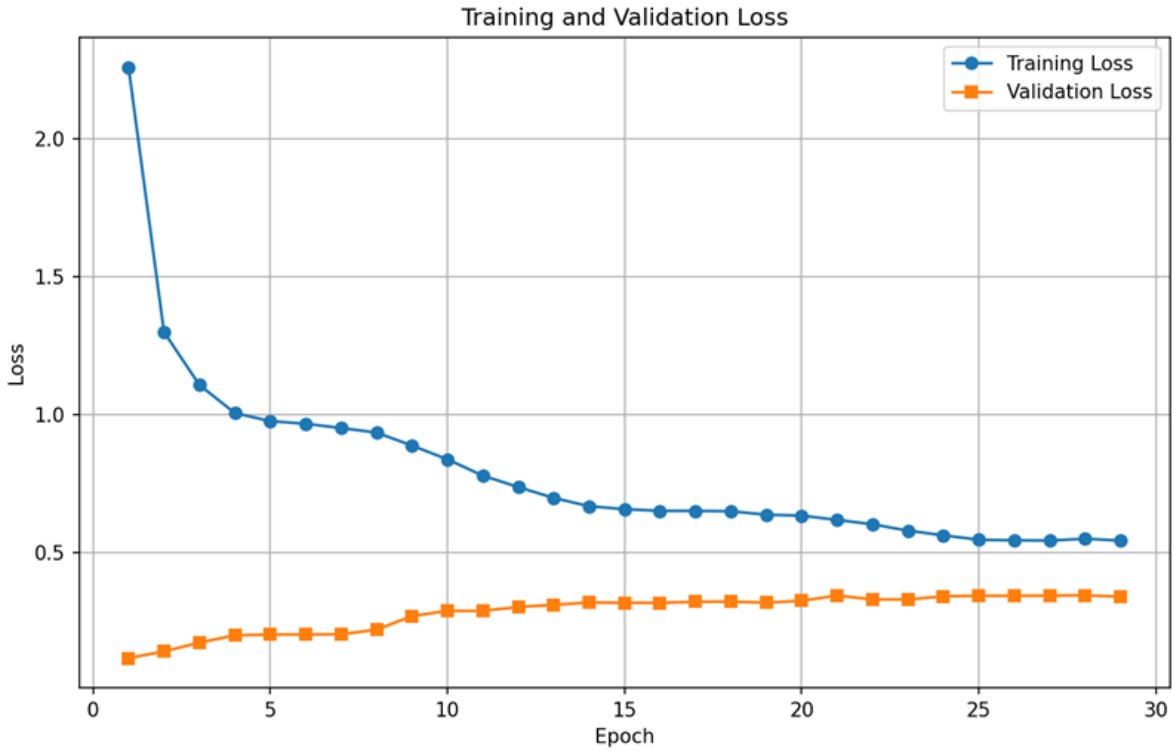


Figure 53: Training-validation loss curves for fine-tuning the DETR model on TSR task with additional parameters.

Figure 53 depicts the training-validation loss curve of fine-tuning the DETR model on the Table structure recognition task with the additional parameters. This model achieved an AP50 of 61.8% on the test dataset and the breakdown of the evaluation results can be seen in Table 4 below.

Table type	Accuracy_con	GriTS_Top	GriTS_Con	GriTS_Loc
Simple tables (19)	77.03%	81.08%	77.03%	69.50%
Complex tables (47)	91.48%	91.80%	91.48%	85.18%
All tables (66)	87.32%	88.72%	87.32%	80.66%
AP50: 61.8%, AP75: 50.3, AP: 43.1, AR: 56.9				

Table 4: Structure recognition results on the test dataset

Table 4 shows that the model performs better on all three GriTS metrics and content accuracy, for complex tables compared to simple tables. Despite a decreasing training loss, the validation loss increases, which implies that the model is overfitting. Overfitting occurs when the model becomes so specialized in the training data, that it loses its ability to generalize well to unseen data.

Hyperparameter	Value
Learning rate	1×10^{-4}
Batch size	8
Weight decay	1×10^{-3}
LR scheduler	Cosine Annealing with $T_{\max} = 50$
LR drop	15
Encoder and decoder layers	6

Table 5: Hyperparameter configuration of the best performing structure recognition model.

To address this issue, the hyperparameters introduced in Section 3.2.5 were optimized sequentially to avoid overfitting the data and to improve the recognition accuracy. The learning rate and Weight Decay parameters were adjusted to find a balance between convergence and model stability. Regularization techniques such as Dropout, discussed in Section 3.2.5, were also introduced to ensure that the model generalizes on unseen data. The network architecture was modified to reduce the model's complexity by decreasing the number of encoder and decoder layers. The batch size was also increased to stabilize training and enhance generalization. Cosine Annealing with Warm Restarts, a cyclic learning rate scheduler that gradually decreases the learning rate following a cosine function, was introduced to periodically reset the learning rates to a higher value to improve convergence and prevent the model from getting stuck in suboptimal solutions. Auxiliary losses were also enabled to provide additional supervision to the model's intermediate

layers during training. Furthermore, early stopping was implemented based on validation loss to stop the training process when the model's performance plateaued. The best hyperparameter configurations for the TSR model are depicted in Table 5.

The fine-tuned model with the optimal hyperparameters achieved an AP50 of 71.3% on the solar module dataset, demonstrating substantial improvement over the pre-trained baseline.

5.3.2 Quantitative Results

The COCO metrics results of various experiments are summarized in the table below:

Model	**AP50**	**AP75**	**AR**	**AR**
Pre-trained model	10.4%	6.8%	6.2%	6.7%
Fine-tuned without canonicalization	9.1%	8.4%	6.9%	7.5%
Fine-tuned with canonicalization	61.8%	50.3%	43.1%	56.9%
Fine-tuned with HPO	71.3%	57.3%	49.0%	58.3%

Table 6: Table Structure Recognition performance comparison on COCO metrics.

From Table 6, it can be observed that COCO metrics only focus on how accurately the table structures are being detected based on their predicted bounding boxes, but now a more detailed approach is required for evaluating table structure at the cellular level. Therefore, GriTS was introduced to capture this finer level of detail by evaluating whether the model correctly identifies and organizes all the cells within a table. The GriTS results can be observed in Table 7. The results indicate that the Fine-Tuned model with HPO achieved a content accuracy of 91.89% and a Topological GriTS score of 93.40%, highlighting the TSR model's ability to correctly identify and organize table structures.

Model	**Table type**	**Acc_con**	**GriTS_Top**	**GriTS_Con**	**GriTS_Loc**
Fine-tuned with canonicalization	Simple tables (19)	77.03%	81.08%	77.03%	69.50%
	Complex tables (47)	91.48%	91.80%	91.48%	85.18%
	All tables (66)	87.32%	88.72%	87.32%	80.66%
Fine-Tuned with HPO	Simple tables (19)	83.21%	89.18%	83.21%	77.13%
	Complex tables (47)	95.03%	94.93%	95.03%	92.18%
	All tables (66)	91.89%	93.40%	91.89%	88.18%

Table 7: Structure evaluation results on the test dataset.

5.3.3 Visualization Results

Figure 54 and Figure 55 demonstrate the output of the TSR model applied to a horizontal and dual-axis table respectively. Table 8 provides the quantitative evaluation results for these figures.

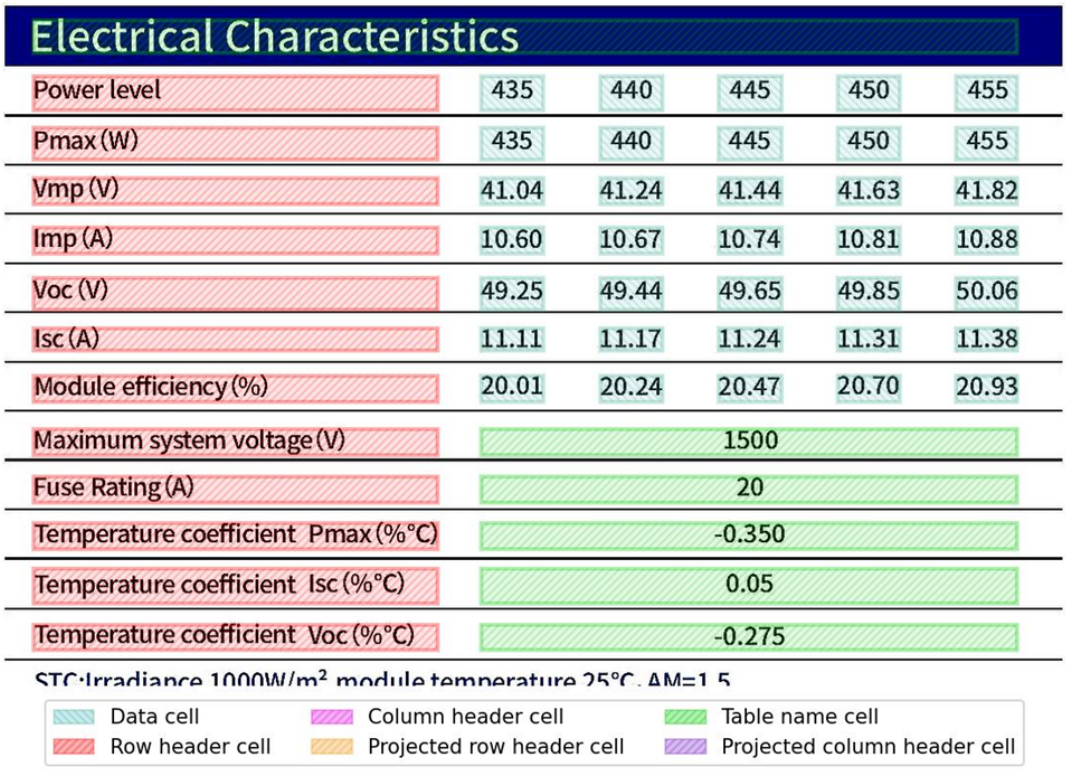


Figure 54: Table structure recognition results on a horizontal table.

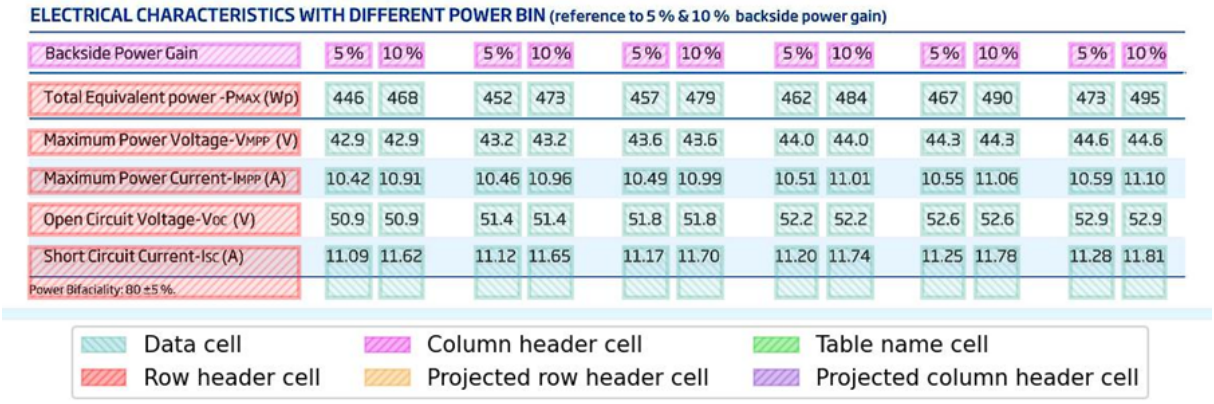


Figure 55: Table structure recognition results on a dual-axis table.

In Figure 54, the model successfully detects the table structures and identifies merged cells. However, the model struggled to distinguish between the header and data cells that span multiple columns. This suggests that the model struggles with the semantic differentiation of structural

elements. In Figure 55, the model processes a dual-axis table. Although the model correctly recognizes the table layout, additional information was captured in the last row. From Table 8, it can be noted that Figure 54 achieves perfect GriTS and high accuracy scores. In contrast, Figure 55 shows lower performance, particularly in AP75 (47.1%) indicating increased complexity in handling multi-header structures.

Image	AP50	AP75	AP	AR	GriTS_Top	GriTS_Con	GriTS_Loc	Acc_Con
Figure 54	92.6%	71.0%	83%	78.7%	1.0	1.0	1.0	1.0
Figure 55	89.2%	47.1%	63.4%	74.3%	0.9091	0.9091	0.9023	0.8334

Table 8: Table structure recognition metrics for Figure 54 and Figure 55

5.4 Experiment - Tabular Data Extraction

This section will provide the experimental setup and present the evaluation results for extracting tabular data from the detected tables. It also includes an analysis of the challenges faced, solutions implemented, and performance metrics achieved during the experimentation.

5.4.1 Experiment Setup

The tabular data extraction process refines recognized table components, such as rows, columns, and headers, into a consistent structure, enabling accurate segmentation and labeling of content. This stage included canonical cell merging for spanning cells, header and data cell classification, row-column mapping, OCR-based cell content retrieval, and exporting the processed tables in formats like CSV and Excel which were already discussed in detail in Section 4.6.

5.4.2 Results

The performance of this experiment was evaluated mainly based on the accuracy of the extracted content. The ground truth data required for evaluating this pipeline was created by manually extracting the data from all the tables in the PV module test set, created in Section 4.1.4, and saving them as YAML files.

Figure 56 depicts an example of a table whose structure was recognized using TATR. The TDE process leveraged the recognized table structures and the enhanced canonicalization algorithm to output a simple table that is logically consistent by concatenating the data in the header cell with that of its sub-headers as seen in Figure 57. This pipeline achieved an average precision of

41.98% and an average recall of 45.58%. The poor extraction performance was mainly attributed to the deficiencies of the OCR as observed in Figure 57.

Electrical Parameters

Module Type	SPICN6(LAR)-60-375/IH		SPICN6(LAR)-60-380/IH		SPICN6(LAR)-60-385/IH	
	STC	NOCT	STC	NOCT	STC	NOCT
Maximum Power (Pmax/W)	375	282	380	286	385	290
Maximum Power Voltage (Vmpp/V)	35.1	32.5	35.3	32.7	35.5	32.9
Maximum Power Current (Impp/A)	10.69	8.68	10.77	8.75	10.85	8.82
Open Circuit Voltage (Voc/V)	41.6	39.8	41.7	39.9	41.8	40.0
Short Circuit Current (Isc/A)	11.50	9.23	11.61	9.32	11.72	9.42
Module Efficiency	20.9%		21.2%		21.5%	

Figure 56: Table structure recognized using DETR containing merged cell in the header.

	A	B	C	D	E	F	G
1	Module Type 20.0kg	SPICN6(LAR)}-60-375/IH | SIC	SPICN6(LAR)}-60-375/IH | NOCT	SPICN6(LAR)-60-380/1H | STC	SPICN6(LAR)-60-380/IH | NOCT	SPICN6(LAR)-60-385/IH | S1E	SPICN6(LAR)-60-385/IH | NOCT
2							
3	Maximum Power (Pmax/W)	375	282	380	286	385	290
4							
5	Maximum Power Voltage (Vmpp/V)	Ee|	S25	35:03:00	32.7	35:05:00	329
6							
7	Maximum Power Current (Impp/A)	10.69	8.68	10.77	875	10.85	8.82
8							
9	Open Circuit Voltage (Voc/V)	41.6	39.8	ALT	39.9	A18	40
10							
11	Short Circuit Current (Isc/A)	11.5	9.23	11.61	9.32	11.72	9.42
12							
13	Module Efficiency	20.90%	20.90%	21.20%	21.20%	21.50%	21.50%

Figure 57: Data extracted from Figure 56.

	A	B	C	D	E	F	G
1	Module Type 20.0kg	SPICN6(LAR)}-60-375/IH | SIE	SPICN6(LAR)}-60-375/IH | NOCT	SPICN6(LAR)-60-380/IH | STC	SPICN6(LAR)-60-380/IH | NOCT	SPICN6(LAR)-60-385/IH | STC	SPICN6(LAR)-60-385/IH | NOCT
2							
3	Maximum Power (Pmax/W)	375	282	380	286	385	290
4							
5	Maximum Power Voltage (Vmpp/V)	35.1	32.5	35.3	32.7	35.5	32.9
6							
7	Maximum Power Current (Impp/A)	10.69	8.68	10.77	8.75	10.85	8.82
8							
9	Open Circuit Voltage (Voc/V)	41.6	39.8	41.7	39.9	41.8	40
10							
11	Short Circuit Current (Isc/A)	11.5	9.23	11.61	9.32	11.72	9.42
12							
13	Module Efficiency	20.90%	20.90%	21.20%	21.20%	21.50%	21.50%

Figure 58: Data extracted from Figure 56 after image and OCR improvements.

The OCR engine often misrecognized text on documents that used non-standard fonts or had degraded document quality. It also extracted extra or incorrect text from densely packed tables, datasheets with unconventional layouts, datasheets where text overlapped graphical elements, and due to poor contrast with the background. The inaccurate bounding boxes around the detected element by the DETR model provided further challenges. To mitigate these issues, the quality of the input image was improved using the techniques described in Section 4.2.2, and the OCR

extraction quality was improved as discussed in Section 4.2.3. Figure 58 shows the improved results after incorporating these changes.

This pipeline was then evaluated on the PV module test dataset. After incorporating image and OCR enhancements, it achieved an average precision of 61.1% and an average recall of 65.67%, showing a significant improvement in the accurate placement of the extracted text within the table cells according to the ground truth YAML files.

Figure 59 depicts a table with a complex structure containing merged cells in the table content and Figure 60 presents the TDE output. It can be noted that the data present in the merged cells are appropriately extracted across the individual cells it spans.

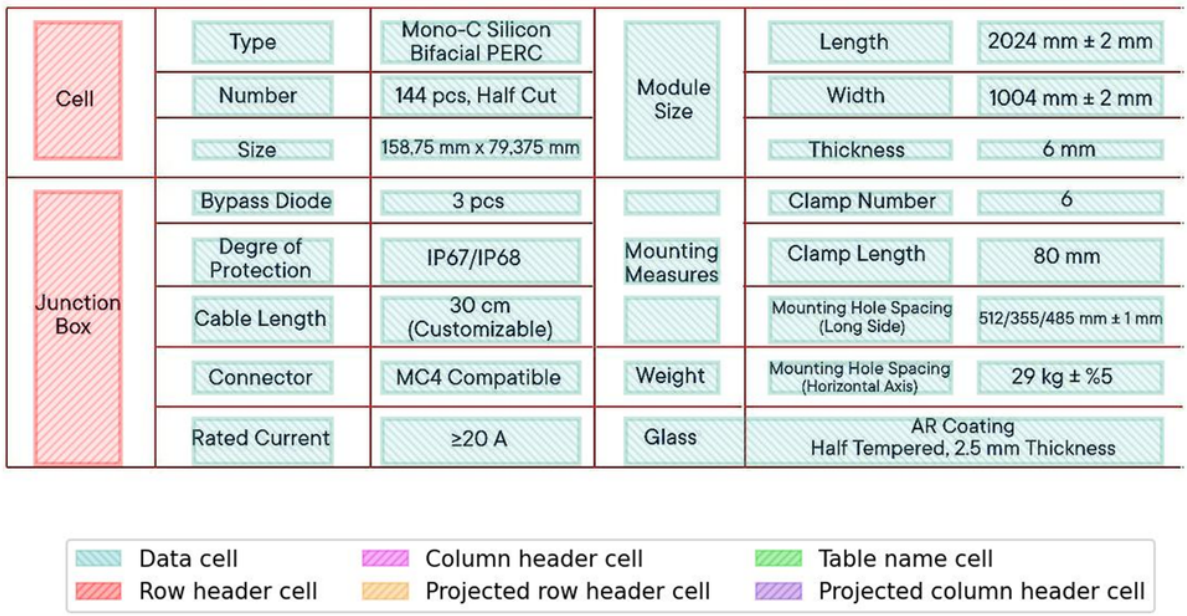


Figure 59: Table structure recognized using DETR containing merged cell in the table content.

	A	B	C	D	E	F
1						
(a) 2	Cell	Type	Mono-C Silicon Bifacial PERC	Moats	Length	2024 mm 2mm
3	Cell	Number	144 pes, Half Cut	Moats	Width	1004 mm 2mm
4	Cell	Size	158,75 mm x 79,375 mm	Moats	Thickness	6mm
5	Juncti Ba	Bypass Diode	3 pes		Clamp Number	6
6	Juncti Ba	Degre of Protection	IP67/IP68	Mounting Measures	Clamp Length	80mm
7	Juncti Ba	Cable Length	iustone stk e)		it Mounting Hae opacing	512/355/485 mm £1 mm
8	Juncti Ba	Connector	MC4 Compatible	Weight	i	29 kg +%5
(b) 9	Juncti Ba	Rated Current	220A	AR Coating Glass Half Tempered, 2.5 mm Thickness	AR Coating Glass Half Tempered, 2.5 mm Thickness	AR Coating Glass Half Tempered, 2.5 mm Thickness

Figure 60: Data extracted from Figure 59 containing merged cells in the table content.

5.5 Experiment - Complete Pipeline Evaluation

Chapter 4 discussed the end-to-end tabular data extraction pipeline that takes a PDF document and outputs the values of interest. This section will provide the experimental setup and present the evaluation results of the complete pipeline, including the results of intermediate post-processing techniques.

5.5.1 Experiment Setup

Dataset Selection

Ten PDF documents with diverse layouts and table styles were selected to evaluate the complete pipeline. These documents were chosen to cover challenges such as irregular table boundaries, complex headers, and dual-axis structures.

For each document, the fields of interest, such as electrical, thermal, and packaging properties, were manually extracted and saved in a human-readable YAML format. This ground truth serves as the benchmark for evaluating the pipeline.

Experiment Breakdown

The evaluation was done by inputting the selected PDF documents into the pipeline. The workflow involved:

- *Preprocessing Datasheets :* As mentioned in Section 4.2, the PDF documents were preprocessed to prepare input images for the pipeline.
- *Table Detection :* The tables were extracted from the PDF images as discussed in Section 4.3.
- *Table Structure Recognition :* The rows, columns, headers, and spanning cells within each detected table were identified, as mentioned in Section 4.4.
- *Tabular Data Extraction :* The raw data was extracted from the structured tables as briefed in Section 4.6.
- *Post-Processing and Final Extraction Step :* Utilizing ML-based algorithms and regular expressions the values of interest were extracted from the structured data as mentioned in Section 4.7.

The experimental setup of the intermediate steps for post-processing is discussed in detail below.

Table Classification

Section 4.7.1 discussed the detailed approach implemented for classifying tables using a TF-IDF word vectorizer and a Naive Bayes classifier. The dataset used for training and testing the table classification model was created from 75 randomly selected PDF files which consists of 266 tables. The raw tabular data was then processed to extract the word tokens which were then classified among the six classes manually and assigned the corresponding class label. Depending on the dataset, the six classes are classified as: *'Electrical Characteristics at Standard Testing Conditions(STC)'*, *'Electrical Characteristics at Nominal Module Operating Temperature (NMOT)'*, *'Thermal Characteristics'*, *'Mechanical Characteristics'*, *'Packaging Characteristics'*, and *'Other '*. The dataset was divided into an 80:20 ratio to create the training and testing sets for the Multinomial Naive Bayes classifier. Table 9 depicts the class-wise performance of the Naive Bayes classifier.

Class	**Precision**	**Recall**	**F1-Score**
Electrical Characteristics at Standard Testing Conditions (STC)	0.75	0.60	0.67
Electrical Characteristics at Nominal Module Operating Temperature (NMOT)	0.33	0.67	0.44
Thermal Characteristics	1.00	0.88	0.93
Mechanical Characteristics	1.00	1.00	1.00
Packaging	0.80	1.00	0.89
Others	1.00	0.94	0.97

Table 9: Per-class evaluation performance of the Naive Bayes classifier with the TF-IDF.

From Table 9, the very low precision values for the *'Electrical Characteristics at Nominal Module Operating Temperature (NMOT)'* class indicates a large number of false positives. This is likely due to the similarity between the features of this class with that of *'Electrical Characteristics at Standard Testing Conditions (STC)'* class. Both the classes have near identical word vectors/tokens with the only difference being the testing conditions. As a result, the model struggles in accurately classifying these tables. This model achieved an overall accuracy of 87% which indicates a robust performance.

Table Orientation Detection

Section 4.5.1 discussed the detailed approach of identifying the dominant axis of a table. The dataset used for testing the orientation of a table was created from 8 randomly selected PDF files

which consist of 25 tables. The raw tabular data was extracted in a structured format and its orientation was determined manually and labeled appropriately.

The structured tables from Section 5.3.1 were used as input for table orientation detection. This process separates numerical data from text and calculates variance along vertical and horizontal axes to determine the table's dominant orientation or classify it as dual-axis if the variance difference is minimal. This process achieved an accuracy of 92% in determining the orientation of a table.

5.5.2 Results

This experiment evaluates the pipeline's ability to accurately extract the values of interest from the PV module datasheets. This end-to-end pipeline achieved an accuracy of 52.66% in extracting the values of interest from the PV module test dataset mentioned in Section 5.5.1 consisting of 10 PDF files. The poor extraction performance was mainly attributed to the deficiencies in the OCR and the inability of the regular expression to handle variations in data provided by different manufacturers.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1		name	year	length	width	E/eff	E/pmpp	E/vmpp	E/impp	E/voc	E/isc	EN/pmpp	EN/vmpp	EN/impp	EN/voc	EN/isc	T/isc	T/pmpp	T/voc	/pcs_palle	/pallet_con
2	0	5-VDS	2023	2095	1039	19.7	430	40.5	10.62	48.7	11.20	325.0	38.2	8.51	46.0	9.02	0.040	-0.34%/°	-0.25%/°	35	770
3	1	5-VDS	2023	2095	1039	20.0	435	40.8	10.67	48.9	11.29	329.0	38.5	8.55	46.2	9.03	0.040	-0.34%/°	-0.25%/°	35	770
4	2	5-VDS	2023	2095	1039	20.2	440	41.1	10.71	49.1	11.37	333.0	38.8	8.58	46.4	9.08	0.040	-0.34%/°	-0.25%/°	35	770
5	3	5-VDS	2023	2095	1039	20.4	445	41.4	10.75	49.3	11.45	337.0	39.0	8.63	46.6	9.12	0.040	-0.34%/°	-0.25%/°	35	770
6	4	5-VDS	2023	2095	1039	20.6	450	41.7	10.80	49.5	11.53	341.0	39.1	8.71	46.7	9.15	0.040	-0.34%/°	-0.25%/°	35	770

Figure 61: Relevant data extracted manually from Figure 52.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1		name	year	length	width	E/eff	E/pmpp	E/vmpp	E/impp	E/voc	E/isc	EN/pmpp	EN/vmpp	EN/impp	EN/voc	EN/isc	T/isc	T/pmpp	T/voc	/pcs_palle	/pallet_con
2	0	5-VDS	2023	2095	1039		457	40.8	11.20	49.0		325.0	38.2	8.51	46.0	9.02	0.040%/°	-0.34%/°	-0.25%/°		770
3	1	5-VDS	2023	2095	1039		479	40.8	11.74	49.1		329.0	38.5	8.55	46.2	9.05	0.040%/°	-0.34%/°	-0.25%/°		770
4	2	5-VDS	2023	2095	1039		500	40.8	12.27	49.2		333.0	38.8	8.58	46.4	9.08	0.040%/°	-0.34%/°	-0.25%/°		770
5	3	5-VDS	2023	2095	1039		522	40.8	12.80	49.3		337.0	39.0	8.63	46.6	9.12	0.040%/°	-0.34%/°	-0.25%/°		770
6	4	5-VDS	2023	2095	1039		544	40.8	13.34	49.4		341.0	39.1	8.71	46.7	9.15	0.040%/°	-0.34%/°	-0.25%/°		770

Figure 62: Relevant data extracted from Figure 52 using this pipeline.

Figure 61 depicts the relevant data that was manually extracted from the PV module datasheet observed in Figure 52. On the other hand, Figure 62 demonstrates the data extracted from Figure 52 using the current pipeline which achieved a precision of 53.2% and a recall of 54.2%. The poor performance in this case can mainly be attributed to TD model's inability to detect the 'Electrical Characteristics table at Standard Testing Conditions (STC)' as seen in Section 5.2.3 and subsequently, the red bounding box in Figure 62, denotes the values captured from an unrelated table. Additionally, some values are missing due to the inability of regular expression to capture all textual variations and minor OCR errors.

6 Discussion and Future Work

This chapter discusses the key findings from the experiments within the context of the thesis. Section 6.1 compares the proposed pipeline with the rule-based Lightning-Table approach, focusing on qualitative evaluations. Section 6.2 explores the strengths and limitations of this pipeline and finally, Section 6.3 suggests improvements to enhance the pipeline's accuracy and adaptability.

6.1 Ablation Studies

Ablation studies were performed to compare the performance of the proposed pipeline with the existing Lightning-Table approach discussed in Section 2.3. The main objective of this study was to assess the pipeline's ability to accurately extract tabular data from PV module datasheets which often contain complex table structures. Only the qualitative aspects of the model's ability to recognize the structure and extract meaningful data from tables were measured since the quantitative analysis was not feasible due to the lack of a common basis for comparison. Even with a test dataset comprising of only text-based PDF files, a fair comparison is not possible since both these pipelines handle merged cells and headers differently and result in varying table layouts.

The Lightning-Table approach discussed in Section 2.3 utilized rule-based methods and heuristics combined with open-source tools such as Camelot [15] and Tabula [30] to identify table boundaries and the internal table components. While effective for structured and well-delineated tables, it encountered significant difficulties with complex layouts, missing grid lines, and rotated tables. Camelot also assumed consistent column spacing and clear row separators, and hence it struggled in processing multi-row/multi-column headers, merged cells, and variable column spans, leading to frequent misalignment and missing values. Figure 63 presents the data extracted from Figure 56 using the Lightning-Table approach. This approach fails to recognize the internal table components such as headers and merged cells. The rule-based approach prioritizes text alignment for data extraction, and as a result, it is unable to map the *'Module Type'* parameter with the corresponding data cells denoted in the bounding boxes. In the post-processing stage, the extracted data is processed either horizontally or vertically, and mapping the *'Module Type'*

to the respective data becomes challenging due to text misalignment. This results in a significant loss of information during the relevant data extraction.

The proposed pipeline addresses these limitations by leveraging a Deep Learning model with self-attention to identify relationships between table elements. This is observed in Figure 58, where the proposed model successfully detects the table headers and merged cells. The headers are then merged together in a way that simplifies the table structure while maintaining the logical flow of data. Although this approach is prone to character misrecognition errors by the OCR engine, the model can still map the *'Module Type'* with the respective data.

	A	B	C	D	E	F	G	H	I	J
1	Module Type		SPICN6(LAR)-60- 375/IH			SPIC N6(LAR)-60- 380/IH			SPICN6(LAR)- 60-385/IH	
2										
3		STC		NOCT	STC		NOCT	STC		NOCT
4										
5	Maximum Power (Priax/W)	375		282	380		286	385		290
6										
7	Maximum Power Voltage (Vmpp/V)	35.1		32.5	35.3		32.7	25.5		32.9
8										
9	Maximum Power Current (Impp/A)	10.69		8.68	10.77		875	10.85		8.82
10										
11	Open Circuit, Voltage (Voc/V)	41.6		39.8	417		39.9	41.8		40.0
12										
13	Short Circuit Current (Isc/A)	11.50		9.23	11.61		9.32	172		9.42
14										
15	Module Efficiency		20.9%			21.2%			21.5%	
16										

Figure 63: Data extracted from Figure 56 using the Lightning-Table pipeline.

	A	B	C	D	E	F	G
1	ElectricalCharacteristics						
2	Powerlevel		435	440	445	450	455
3	Pmax （W）		435	440	445	450	455
4	Vmp （V）		41.04	41.24	41.44	41.63	41.82
5	Imp （A）		10.60	10.67	10.74	10.81	10.88
6	Voc （V）		49.25	49.44	49.65	49.85	50.06
7	Isc （A）		11.11	11.17	11.24	11.31	11.38
8	Moduleefficiency （%）		20.01	20.24	20.47	20.70	20.93
9	Maximumsystemvoltage （V）				1500		
10	FuseRating （A）				20		
11	Temperaturecoefficient	Pmax （%℃）			-0.350		
12	Temperaturecoefficient	Isc （%℃）			0.05		
13	Temperaturecoefficient	Voc （%℃）			-0.275		

Figure 64: Data extracted from Figure 54 using the Lightning-Table pipeline.

Figure 64 illustrates the data that was extracted from Figure 54 using the Lightning table approach. Although the data extraction is accurate, with no misrecognized characters, the model's inability to recognize merged cells results in missing values. In contrast, Figure 65 depicts the data extracted from Figure 54 using the proposed pipeline where the content of merged cells

is appropriately extracted across the individual cells it spans. From Figure 65, we observe that TATR captures the data accurately and ensures that there is no loss of information. However, character misrecognition by OCR can result in faulty and unreliable data extraction.

	A	B	C	D	E	F	G
1	Electrical Characteristics	Electrical (	Electrical (	Electrical (	Electrical (	Electrical Characteristics	
2	Power level	435	440	445	450	455	
3	Pmax (W)	435	440	445	450	455	
4	Vmp (V)	41.04	4124	4144	4163	41.82	
5	Imp (A)	10.60	10.67	10.74	10.81	10.88	
6	Voc (V)	49.25	49.44	49.65	49.85	50.06	
7	Isc (A)	11.11	11.17	11.24	11.31	11.38	
8	Module efficiency (%)	20.01	20.24	20.47	20.70	20.93	
9	Maximum system voltage (V)	1500	1500	1500	1500	1500	
10	Fuse Rating (A)	20	20	20	20	20	
11	Temperature coefficient Prax (%°C)	-0.350	-0.350	-0.350	-0.350	-0.350	
12	Temperature coefficient Isc (%°C)	0.05	0.05	0.05	0.05	0.05	
13	Temperature coefficient Voc (%°C)	-0.275	-0.275	-0.275	-0.275	-0.275	

Figure 65: Data extracted from Figure 54 using the current pipeline.

In summary, the proposed pipeline offers significant improvements over the Lightning-Table approach in handling complex table structures, effectively recognizing internal components, and ensuring no loss of information during data extraction.

6.2 Discussion

The proposed approach addressed several limitations observed in the Lightning-Table pipeline elaborated in Section 2.3, which struggled in extracting data from tables with complex structures, image-based documents, fragmented entries, and inconsistent text alignment. Tables present in PV module datasheets often have merged data cells, headers spanning multiple rows/columns, or unconventional layouts. However, Lightning-table relied on rule-based extractions only from text-based PDFs and faced significant challenges while parsing multi-line cells or text with variable spacings and alignment as seen previously in Figure 16 and Figure 17 respectively.

In contrast, the proposed approach recognizes the relationships between rows and columns while also capturing the contextual and structural information more accurately. This also includes correctly mapping columns with varying widths or merging cells that span multiple rows. By using a transformer-based table recognition model along with optical character recognition (OCR),

the multimodal approach proposed in this thesis achieves improved recognition of table structures and in turn a higher data extraction quality.

Incorporating OCR for data extraction as mentioned in Section 4.2.3, extends this pipeline's compatibility with both text-based and image-based PDFs. The conversion of PDF documents to images discussed in Section 4.2.1 standardizes the input format and enables the datasheets to be processed visually and also eliminates the reliance on the internal PDF structure that frequently differs across manufacturers. Furthermore, this approach preserves graphical elements and ensures visual consistency thereby enhancing the accuracy of the DETR models.

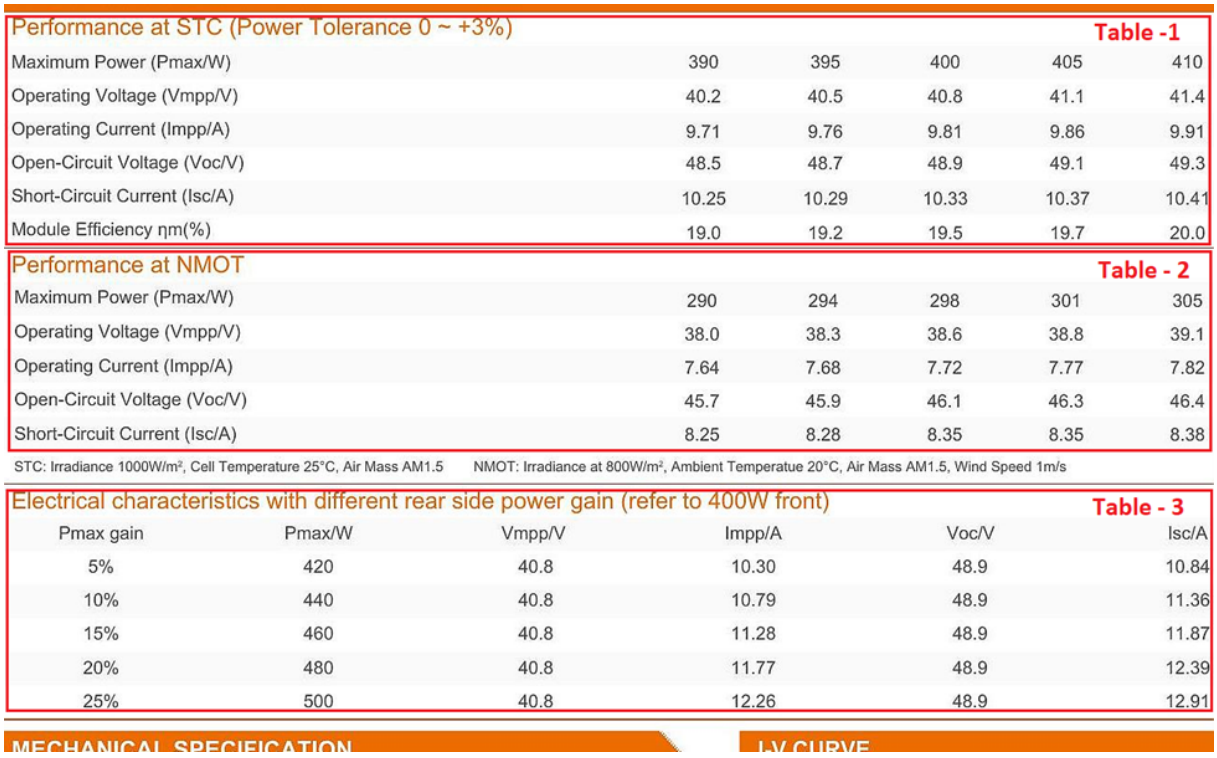
The DETR transformers used for Table Detection and Table Structure Recognition were explicitly fine-tuned on tables with complex structures and densely packed tables like datasheets to identify table regions and their components with high precision. Furthermore, the improved canonicalization algorithm introduced in Section 4.4.3 further advanced this pipeline and efficient handling of dual-axis tables. By systematically processing the recognized table elements, this algorithm mitigates the over-segmentation problem and converts complex tables into simpler structures for efficient data extraction.

Despite these enhancements, several challenges still exist. One major challenge during model training pertains to the complexity in annotating the datasheets during the data preparation stage discussed in Section 4.1.2. The manual labeling of table structures can be time consuming, labor-intensive, and might require additional domain knowledge, particularly in annotating PV datasheets with multiple header rows or complicated layouts. This approach introduced slight padding variations and alignment mismatch while annotating the table elements, especially with varying manufacturer styles and branding schemes. Class imbalance in training data further compounds the issue, as some table elements, such as spanning cells, projected row headers, or rotated tables, appear less frequently. As a result, the trained model exhibits bias or lower accuracy when encountering underrepresented classes during inference.

Extracting and processing high resolution images of the datasheet can be computationally expensive. On the other hand, low-resolution input datasheets or scanned datasheets containing artifacts hinder the OCR engine's ability to accurately recognize text. Although image enhancement techniques described in Section 4.2.2 help mitigate these problems, non-standard fonts, faint texts, densely packed content, poor contrast, engine limitations, and character ambiguities often lead to misrecognition. Overlapping graphical elements and watermarks were also observed in some datasheets which can obscure or distort important information. These factors reduce OCR

performance and introduce errors that require subsequent validation steps. Correcting such errors often requires manual intervention or advanced techniques for context-based validation.

The Table Detection model discussed in Section 5.2 was robust in ensuring that the structure recognition and further data extraction was applied only to meaningful content and thereby improved efficiency by reducing the computational overhead. However, it faced hurdles when dealing with densely packed tables or nested structures with irregular layouts. Figure 66 presents the image of a table that was detected by the model, but it actually consists of three separate tables as highlighted by the bounding box. This issue arises due to the model's inability to accurately localize tables in close proximity. Low-quality input images, complex backgrounds, or embedded icons also hindered the accurate prediction of bounding boxes. Although OCR provides spatial data regarding the content, it often leads to false positives as the models misclassify other structured information like graphs as tables.



Performance at STC (Power Tolerance 0 ~ +3%) — Table -1

Maximum Power (Pmax/W)	390	395	400	405	410
Operating Voltage (Vmpp/V)	40.2	40.5	40.8	41.1	41.4
Operating Current (Impp/A)	9.71	9.76	9.81	9.86	9.91
Open-Circuit Voltage (Voc/V)	48.5	48.7	48.9	49.1	49.3
Short-Circuit Current (Isc/A)	10.25	10.29	10.33	10.37	10.41
Module Efficiency ηm(%)	19.0	19.2	19.5	19.7	20.0

Performance at NMOT — Table - 2

Maximum Power (Pmax/W)	290	294	298	301	305
Operating Voltage (Vmpp/V)	38.0	38.3	38.6	38.8	39.1
Operating Current (Impp/A)	7.64	7.68	7.72	7.77	7.82
Open-Circuit Voltage (Voc/V)	45.7	45.9	46.1	46.3	46.4
Short-Circuit Current (Isc/A)	8.25	8.28	8.35	8.35	8.38

STC: Irradiance 1000W/m², Cell Temperature 25°C, Air Mass AM1.5 NMOT: Irradiance at 800W/m², Ambient Temperatue 20°C, Air Mass AM1.5, Wind Speed 1m/s

Electrical characteristics with different rear side power gain (refer to 400W front) — Table - 3

Pmax gain	Pmax/W	Vmpp/V	Impp/A	Voc/V	Isc/A
5%	420	40.8	10.30	48.9	10.84
10%	440	40.8	10.79	48.9	11.36
15%	460	40.8	11.28	48.9	11.87
20%	480	40.8	11.77	48.9	12.39
25%	500	40.8	12.26	48.9	12.91

MECHANICAL SPECIFICATION I-V CURVE

Figure 66: A table detected and cropped by the TD model. The whole image was detected as a table, but it actually comprises of three tables as highlighted by the bounding box.

Since manual annotations were used for model training and ground truth creation, any minor discrepancies between manual annotations and model predictions can affect the intersection-over-union (IoU) scores and lower the performance metrics even when the results look visually correct as seen in Figure 51. This illustrates the sensitivity of the detection metrics at higher thresholds and highlights the importance of consistent annotation protocols.

Variation in layout and irregular table designs complicates the extraction of consistent row-column mappings, especially with multiple headings/subheadings or when merged cells are present in the table. Performance also degrades when the bounding boxes capture incomplete segments of text, which causes faulty merging or splitting of cells. These challenges hinder the accuracy of the table structure recognition model described in Section 5.3.

In summary, the improved approach successfully addresses many shortcomings of the Lightning-table pipeline, including the handling of complex tables, image-based documents, and merged cells. However, it still encounters problems with annotation complexity, class imbalance, noisy inputs, and OCR errors that degrade text recognition. Table detection and Table structure recognition accuracy can also drop in densely packed or irregular layouts, which in turn reduces the data extraction quality. These issues highlight the importance of further research and refinements to create a fully automated system capable of accurately processing a wide variety of PV datasheets.

6.3 Future Work

From the above findings and critical examination of the results from the thesis experiments, the following strategies can be implemented to overcome the aforementioned drawbacks and improve the pipeline's capabilities:

- **Enhanced Dataset Diversity :** Expanding the training dataset to include datasheets from other manufacturers, with different branding schemes, consisting of diverse table layouts, colors, and fonts, as well as tables with various orientations, and structures can improve model generalization. This can reduce the bias and help the model handle diverse datasheet schemes with extremely stylized content.
- **Synthetic Data Generation :** Augmenting the training dataset by artificially generating tables with sparsely found table elements can help mitigate the class imbalance issue. These artificially generated tables with merged cells, rotated text, or unusual row-column spans can help the model learn to handle underrepresented features more effectively.
- **Improving the Detection Issue :** The table detection issue indicated in Figure 66 could be addressed by applying post-processing techniques to validate large bounding boxes based on the internal table structures or by introducing additional training samples with similar densely-packed table layouts.

- **Integration of Advanced OCR Techniques :** Employing cutting-edge OCR engines or DL-based text recognition models, such as CRNN [48] (Convolutional Recurrent Neural Network For Text Recognition) or hybrid transformer architectures, may reduce errors caused by noise and non-standard fonts. This could include training custom OCR modules tailored to fonts and layouts typically found in solar datasheets.
- **Improving Post-Processing steps :** Incorporating stronger dictionary-based correction and context-aware validation rules would help eliminate unnecessary characters and misclassifications.
- **Refined Bounding Box Accuracy :** Additional fine-tuning of the detection model on graphically dense or cluttered documents can reduce false positives. Enhanced annotation protocols where bounding boxes follow a standard margin or padding can minimize discrepancies that harm IoU-based performance metrics.
- **Multi-Engine or Ensemble OCR :** Deploying multiple OCR engines in parallel, and then consolidating their outputs, may improve reliability by capturing diverse recognition strengths. The pipeline can then select the most plausible result through confidence-score comparison.
- **Cross-Domain Validation and Transfer Learning :** Evaluating the model on different kinds of documents, such as financial reports or scientific articles, and then using the insights from these cross-domain tests to guide transfer learning strategies will help the model adapt to a variety of tabular data.
- **Use of LLMs Learning :** Although manual testing with GPT-4 which was performed in Section 4.7.2 demonstrated promising qualitative results in handling diverse table structures and misclassifications, its integration into the pipeline was not pursued due to the costs of using proprietary APIs. This highlights the need for cost-effective alternatives or open-source LLMs that can be integrated into this pipeline. Additionally, refining the methodology to better address poorly labeled headers and diverse table formats helps create a robust, automated solution for tabular data extraction using LLMs.

These suggestion would make the pipeline more resilient to complex table layouts, noise artifacts, and OCR inconsistencies. The integration of synthetic data, advanced OCR engines, improved table localization, and layout analysis techniques could further enhance table recognition accuracy. Ultimately, a combination of diverse training dataset coverage, natural language understanding, and improved post-processing techniques will enable a fully automated solution for extracting tabular data from diverse, real-world PV module datasheets.

7 Summary and Conclusion

This thesis explored the use of a DL-based approach to extract data from PV module datasheets, which often present crucial information regarding the module's specifications in unstructured formats. Motivated by the growing demand for solar energy consumption in recent times, this thesis addresses the challenges of extracting valuable information from densely-packed PV module datasheets with complex, inconsistent layouts that vary across different manufacturers. Using the Table Transformer model which comprises two Detection Transformers, the proposed pipeline can detect tables, recognize their structure, extract the data, and convert them into easily structured formats for compatibility with existing data extraction tools for further analysis. This multi-modal approach of using both the PDF images and OCR tokens, enabled the model to better identify headers, merged cells, and multi-level structures, significantly improving the extraction accuracy. These advancements enable new opportunities for faster, reliable, and accurate data processing that enables researchers and industry stakeholders to better compare, develop, and optimize solar technologies.

Several challenges emerged during the development and testing of this pipeline. Firstly, the lack of a domain-specific dataset led to the manual creation of a labeled dataset for training the model. This process was labor intensive, required domain expertise, and introduced slight inconsistencies/variations in the ground truth data, which significantly affected the model's performance. Furthermore, the small and unbalanced dataset created by this process, and sparse appearance of certain table elements risked underfitting. These drawbacks led to the implementation of data augmentation methods. Secondly, the OCR data extraction errors in documents with low resolution or poor contrast disrupted the performance of the Table Structure Recognition model and required advanced image pre-processing techniques and OCR enhancements. Despite these challenges, evaluations demonstrate that the pipeline achieves robust tabular data extraction performance on various PV datasheets, providing stakeholders with structured data for large-scale data analyses. Additionally, the domain-specific training ensured that the solar-industry terms and parameters were handled effectively. Nevertheless, the final data validation and extraction process produced sub-optimal results. This paves the way for further research into data-driven approaches to improve the quality of data extraction.

In the future, there is room for improvement by creating a larger and more diverse training set

for training the transformer models. This process would make the model robust and enhance the pipeline's ability to recognize complex table structures and visual variations. Integrating semantic checks or using self-supervised approaches might also reduce dependency on annotated data and allow a wider application across other domains with complex document layouts. The continued refinements and incorporation of the suggested enhancements discussed in Section 6.3 could accelerate solar energy research and support a greener future. On a broader aspect, this pipeline could be used to extract tabular data from densely packed documents in the fields of Healthcare, Finance, Legal, and other domains.

Bibliography

[1] N. M. Haegel, P. Verlinden, M. Victoria, P. Altermatt, H. Atwater, T. Barnes, C. Breyer, C. Case, S. D. Wolf, C. Deline, M. Dharmrin, B. Dimmler, M. Gloeckler, J. C. Goldschmidt, B. Hallam, S. Haussener, B. Holder, U. Jaeger, A. Jaeger-Waldau, I. Kaizuka, H. Kikusato, B. Kroposki, S. Kurtz, K. Matsubara, S. Nowak, K. Ogimoto, C. Peter, I. M. Peters, S. Philipps, M. Powalla, U. Rau, T. Reindl, M. Roumpani, K. Sakurai, C. Schorn, P. Schossig, R. Schlatmann, R. Sinton, A. Slaoui, B. L. Smith, P. Schneidewind, B. Stanbery, M. Topic, W. Tumas, J. Vasi, M. Vetter, E. Weber, A. W. Weeber, A. Weidlich, D. Weiss, and A. W. Bett, "Photovoltaics at multi-terawatt scale: Waiting is not an option," *Science*, vol. 380, no. 6640, pp. 39–42, 2023.

[2] Sunergy USA, *SUN-54MD-H8NS Photovoltaic Module Datasheet*. Sunergy USA, 2024.

[3] RCM Solar, *RCM-XXX-7DBNGXXX410-440 Photovoltaic Module Datasheet*. RCM Solar, 2024.

[4] F. Shafait and R. Smith, "Table detection in heterogeneous documents," in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, DAS '10, (New York, NY, USA), p. 65–72, Association for Computing Machinery, 2010.

[5] G. Harit and A. Bansal, "Table detection in document images using header and trailer patterns," in *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing*, ICVGIP '12, (New York, NY, USA), Association for Computing Machinery, 2012.

[6] 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. 01, pp. 771–776, 2017.

[7] C. Ma, W. Lin, L. Sun, and Q. Huo, "Robust table detection and structure recognition from heterogeneous document images," *Pattern Recognition*, vol. 133, p. 109006, 2023.

[8] S. Paliwal, V. D, R. Rahul, M. Sharma, and L. Vig, "Tablenet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images," 2020.

[9] K. Itonori, "Table structure recognition based on textblock arrangement and ruled line position," in *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR '93)*, pp. 765–768, 1993.

[10] W. Lin, Z. Sun, C. Ma, M. Li, J. Wang, L. Sun, and Q. Huo, "Tsrformer: Table structure recognition with transformers," 2022.

[11] S. Raja, A. Mondal, and C. V. Jawahar, "Visual understanding of complex table structures from document images," 2021.

[12] Z. Chi, H. Huang, H.-D. Xu, H. Yu, W. Yin, and X.-L. Mao, "Complicated table structure recognition," 2019.

[13] M. M. Malik, *Extraction of solar cell data from PDF datasheets.* Apr. 2023.

[14] DMEGC Solar, *GH415M6-B66HST-HBT-C 12mm Photovoltaic Module Datasheet.* DMEGC Solar, 2024.

[15] V. Mehta, "Announcing camelot, a python library to extract tabular data from pdfs," Aug. 2019.

[16] G. Blog, "The evolution and core concepts of deep learning neural networks," July 2020.

[17] S. Saha, "A comprehensive guide to convolutional neural networks — the eli5 way," *Medium*, Apr. 2023.

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

[19] Hatim, "Resnet - understanding the revolutionary neural network architecture for deep learning," Sept. 2021.

[20] A. Malik, "Transfer learning with practical implementation by ankit malik | medium," *Medium*, Nov. 2024.

[21] J. de Nyandwi, "Early stopping explained! | medium." Nov 18, 2021.

[22] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," 2020.

[23] B. Smock, R. Pesala, and R. Abraham, "Pubtables-1m: Towards comprehensive table extraction from unstructured documents," 2021.

[24] Tamesol, *650-670-132HC Photovoltaic Module Datasheet*. Tamesol, 2024.

[25] N. Tomar, "What is intersection over union (iou) in object detection?," Feb. 2023.

[26] Evidently AI Team, "How to interpret a confusion matrix for a machine learning model." Accessed: 2025-01-23.

[27] L. Stone, "The challenges of table data extraction - nrx assethub," Nov. 2024.

[28] Ada, "Tech deep dive: Extraction of table data (and why it's difficult) – extraction part 1," Jan. 2023.

[29] R. Jay, "Table extraction using llms: Unlocking structured data from documents," Jan. 2025.

[30] Tabulapdf, "Github - tabulapdf/tabula: Tabula is a tool for liberating data tables trapped inside pdf files."

[31] S. Chandran and R. Kasturi, "Structural recognition of tabulated data," in *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR '93)*, pp. 516–519, 1993.

[32] Tesseract-Ocr, "Github - tesseract-ocr/tesseract: Tesseract open source ocr engine (main repository)."

[33] S. F. Rashid, A. Akmal, M. Adnan, A. A. Aslam, and A. Dengel, "Table recognition in heterogeneous documents using machine learning," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, pp. 777–782, 2017.

[34] 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. 01, pp. 1162–1167, 2017.

[35] T. G. Kieninger, "Table structure recognition based on robust block segmentation," in *Document Recognition V* (D. P. Lopresti and J. Zhou, eds.), vol. 3305, pp. 22 – 32, International Society for Optics and Photonics, SPIE, 1998.

[36] X. Zheng, D. Burdick, L. Popa, X. Zhong, and N. X. R. Wang, "Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context," 2020.

[37] Wikipedia contributors, "Tf–idf," jan 2025. Accessed: 2025-01-23.

[38] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge: Cambridge University Press, jul 2008.

[39] D. Wood and X. Wang, *Tabular abstraction, editing, and formatting*. PhD thesis, University of Waterloo, jan 1996.

[40] Pymupdf, "Github - pymupdf/pymupdf: Pymupdf is a high performance python library for data extraction, analysis, conversion manipulation of pdf (and other) documents.."

[41] HumanSignal, "Github - humansignal/labelimg: Labelimg is now part of the label studio community. the popular image annotation tool created by tzutalin is no longer actively being developed, but you can check out label studio, the open source data labeling tool for images, text, hypertext, audio, video and time-series data.."

[42] Belval, "Github - belval/pdf2image: A python module that wraps the pdftoppm utility to convert pdf to pil image object."

[43] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Word2vec," 2013. Accessed: 2025-02-05.

[44] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Łukasz Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, J. H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Łukasz Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew,

S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. de Avila Belbute Peres, M. Petrov, H. P. de Oliveira Pinto, Michael, Pokorny, M. Pokrass, V. H. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. B. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, and B. Zoph, "Gpt-4 technical report," 2024.

[45] Rapidfuzz, "Github - rapidfuzz/levenshtein: The levenshtein python c extension module contains functions for fast computation of levenshtein distance and string similarity."

[46] B. Smock, R. Pesala, and R. Abraham, "Grits: Grid table similarity metric for table structure recognition," 2023.

[47] L. Wood and F. Chollet, "Efficient graph-friendly coco metric computation for train-time model evaluation," 2022.

[48] A. Yadav, S. Singh, M. Siddique, N. Mehta, and A. Kotangale, "Ocr using crnn: A deep learning approach for text recognition," *2023 4th International Conference for Emerging Technology (INCET)*, pp. 1–6, 2023.